

**ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**

**FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**

**ESCUELA DE INGENIERÍA EN SISTEMAS**

**“ANÁLISIS COMPARATIVO ENTRE LOS FRAMEWORKS JAVASCRIPT MVC, ANGULARJS Y EMBER JS PARA EL DESARROLLO DE APLICACIONES WEB.**

**CASO PRÁCTICO: “ SISTEMA DE CONTROL DE BOTIQUIN VETERINARIO PARA EL MAGAP, MORONA SANTIAGO”**

Trabajo de titulación presentado para optar al grado de:

**INGENIERO EN SISTEMAS INFORMÁTICOS**

**AUTORES:** JORGE FAVIÁN PEÑA ARPI

MILTON GEOVANNY CAMBISACA SÁNCHEZ

**TUTOR:** ING.ANGEL FLORES OROZCO

Macas, Ecuador

2015

**©2015,** Jorge Favián Peña Arpi, Milton Geovanny Cambisaca Sánchez.

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento, siempre y cuando se reconozca el Derecho de Autor.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jorge Favián Peña Arpi Milton Geovanny Cambisaca Sánchez

**ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**

**FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**

**ESCUELA DE INGENIERÍA EN SISTEMAS**

El Tribunal del trabajo de titulación certifica que: ANÁLISIS COMPARATIVO ENTRE LOS FRAMEWORKS JAVASCRIPT MVC, ANGULARJS Y EMBERJS PARA EL DESARROLLO DE APLICACIONES WEB. CASO PRÁCTICO: SISTEMA DE CONTROL DE BOTIQUÍN VETERINARIO PARA EL MAGAP MORONA SANTIAGO*,* de responsabilidad de los señores Jorge Favián Peña Arpi y Milton Geovanny Cambisaca Sánchez, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación, quedando autorizada su presentación.

DR. GONZALO SAMANIEGO

**DECANO DE LA FACULTAD DE ------------------------- --------------------------**

**INFORMÁTICA Y ELECTRÓNICA**

DR. JULIO SANTILLAN

**DIRECTOR DE ESCUELA DE ------------------------- --------------------------**

**INGENIERÍA EN SISTEMAS**

ING. ANGEL FLORES

**DIRECTOR DE TESIS --------------------------- ---------------------------**

ING. EDUARDO VILLA

**MIEMBRO DE TRIBUNAL -------------------------- ---------------------------**

**DOCUMENTALISTA -------------------------- ---------------------------**

**SISBIB ESPOCH**

“Nosotros JORGE FAVIÁN PEÑA ARPI Y MILTON GEOVANNY CAMBISACA SÁNCHEZ, somos responsables de las ideas y resultados expuestos en este trabajo de titulación; y, el patrimonio intelectual de la misma pertenecen a la Escuela Superior Politécnica de Chimborazo”.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jorge Favián Peña Arpi Milton Geovanny Cambisaca Sánchez

DEDICATORIA

A mi Dios, por haberme dado la oportunidad de vivir, saber guiar mi camino y cuidarme cada día de mi vida. A mis padres y hermanos, por su apoyo, ayuda y el cariño incondicional, cual es la razón para seguir adelante en busca de un mejor futuro.

**Milton**

Dedico este proyecto de tesis a mi Dios, por brindarme de su infinita bondad y misericordia, permitiéndome experimentar de aquellos retos y pruebas disfrazados de las más grandes bendiciones, a mi padre, quien sin serlo ante la ley, asumió este rol, dedicando su apoyo y confianza hasta la culminación de mis estudios .

**Jorge**

AGRADECIMIENTO

Deseo agradecer a Dios, por derramar infinitas bendiciones, acompañándome en los buenos y malos momentos, dándome vida, paciencia, sabiduría, permitiéndome alcanzar mis objetivos. Un agradecimiento sincero y profundo a mis padres, Zoila y Carlos, quienes me han guiado y enseñado a trabajar y luchar para conseguir las metas que me he propuesto.

A todos los docentes de la ESPOCH ext. Morona Santiago, quienes nos han compartido sus conocimientos y experiencias y para así formarnos como profesionales. Y expreso un agradecimiento fraterno a nuestro director de tesis, Ing. Ángel Flores, por su apoyo y colaboración en el desarrollo de esta tesis.

**Milton**

Mi agradecimiento a Dios, por ser el pilar fundamental en cada etapa de mi vida, demostrando su eterno amor en tiempos de felicidad y tristeza, guiando mis pasos hacia el cumplimiento de sueños que se vuelven realidad.

A mis padres, Luis y Rafaela, quienes con gran esfuerzo dedicaron todo en cuanto estuvo a su alcance, enseñándome que la perseverancia y entrega tienen su recompensa. A mis amigos, compañeros y hermanos en la fe, que con sus buenos deseos y oraciones, expresaron su apoyo día a día hasta la culminación de esta tesis.

**Jorge**

TABLA DE CONTENIDO

**PORTADA…………………………………………………………………………………….…i**

**DERECHOS DE AUTOR……………………………………………………………………...ii**

**CERTIFICACION…..…………………………………………………………………………iii**

**DECLARACION DE RESPONSABILIDAD…………,……………………………………..iv**

[DEDICATORIA v](#_Toc433738166)

[AGRADECIMIENTO vi](#_Toc433738167)

[TABLA DE CONTENIDO vii](#_Toc433738168)

[ÍNDICE DE TABLAS xi](#_Toc433738169)

[ÍNDICE DE FIGURAS xiii](#_Toc433738170)

[ÍNDICE DE ANEXOS xv](#_Toc433738171)

[RESUMEN xvi](#_Toc433738172)

[ABSTRACT xvii](#_Toc433738173)

[INTRODUCCIÓN 1](#_Toc433738174)

[CAPÍTULO I](#_Toc433738175)

[1. MARCO TEÓRICO 7](#_Toc433738176)

[1.1. Aplicación Web 7](#_Toc433738177)

[1.1.1. Concepto 7](#_Toc433738180)

[1.1.2. Características 7](#_Toc433738181)

[1.2. Servicios Web 8](#_Toc433738182)

[1.2.1. Definición 8](#_Toc433738183)

[1.2.2. Intervinientes 9](#_Toc433738184)

[1.2.3. Estándares 9](#_Toc433738185)

[1.3. Servicios Web REST 10](#_Toc433738187)

[1.3.1. Definición de REST 10](#_Toc433738188)

[1.3.2. Características de los Servicios Web REST 10](#_Toc433738189)

[1.3.3. Formato JSON 11](#_Toc433738190)

[1.4. Tecnología JavaScript 12](#_Toc433738191)

[1.4.1. Introducción 12](#_Toc433738192)

[1.4.2. ¿Cómo identificar código JavaScript? 12](#_Toc433738193)

[1.4.3. Ventajas 13](#_Toc433738194)

[1.4.4. Desventajas 14](#_Toc433738195)

[1.5. Arquitectura MVC 14](#_Toc433738196)

[1.5.1. Patrón de arquitectura Modelo Vista Controlador (MVC) 14](#_Toc433738197)

[1.5.2. Capa Modelo 15](#_Toc433738198)

[1.5.3. Capa Vista 16](#_Toc433738199)

[1.5.4. Capa Controlador 16](#_Toc433738200)

[1.6. Framework AngularJS 16](#_Toc433738201)

[1.6.1. Introducción 16](#_Toc433738202)

[1.6.2. Instalación 17](#_Toc433738203)

[1.6.3. Conceptos Generales 20](#_Toc433738204)

[1.6.4. Características de AngularJS 21](#_Toc433738205)

[1.6.4.1. Data Binding 21](#_Toc433738206)

[1.6.4.2. Directivas 22](#_Toc433738207)

[1.6.4.3. Expresiones 25](#_Toc433738208)

[1.6.4.4. Módulos 25](#_Toc433738209)

[1.6.4.5. $Scope 26](#_Toc433738210)

[1.6.4.6. Controladores 27](#_Toc433738211)

[1.6.4.7. Servicios 28](#_Toc433738212)

[1.7. Framework EmberJS 28](#_Toc433738213)

[1.7.1. Definición EmberJS 28](#_Toc433738214)

[1.7.2. Características generales de uso 29](#_Toc433738215)

[1.7.3. Implementar EmberJS 29](#_Toc433738216)

[1.7.4. Mi primera Aplicación con EmberJS 30](#_Toc433738217)

[1.7.5. Estructura básica de una aplicación con EmberJS 33](#_Toc433738218)

[CAPITULO II](#_Toc433738219)

[2. MARCO METODOLÓGICO 35](#_Toc433738220)

[2.1. Introducción 35](#_Toc433738221)

[2.2. Definición de Parámetros de comparación 35](#_Toc433738222)

[2.3. Descripción de los módulos de pruebas 37](#_Toc433738223)

[2.4. Desarrollo de los módulos de pruebas 37](#_Toc433738224)

[2.4.1. Framework AngularJS 37](#_Toc433738225)

[2.4.2. Framework EmberJS 38](#_Toc433738226)

[CAPITULO III](#_Toc433738227)

[3. MARCO DE RESULTADOS, DISCUSIÓN Y ANÁLISIS DE RESULTADOS 40](#_Toc433738228)

[3.1. Análisis Comparativo entre los Framework AngularJS y EmberJS 40](#_Toc433738229)

[3.1.1. Indicador 1: CPU 40](#_Toc433738230)

[3.1.2. Indicador 2: Memoria RAM 42](#_Toc433738231)

[3.1.3. Indicador 3: Disco 42](#_Toc433738232)

[3.1.4. Indicador 4: Red 43](#_Toc433738233)

[3.1.5. Indicador 5: Tarjeta Gráfica 43](#_Toc433738234)

[3.2. Análisis de Resultados 45](#_Toc433738235)

[3.3. Interpretación de los resultados 47](#_Toc433738236)

[3.4. Comprobación de la Hipótesis 48](#_Toc433738237)

[CAPITULO IV](#_Toc433738238)

[4. PROPUESTA DE DESARROLLO DEL SISTEMA DE CONTROL DE BOTIQUINES (SICOBO)……………………………………………………….51](#_Toc433738239)

[4.1. Visión y Alcance 51](#_Toc433738240)

[4.1.1. Definición del Problema. 51](#_Toc433738241)

[4.1.2. Perfiles de Usuario. 52](#_Toc433738242)

[4.1.3. Ámbito del Proyecto. 53](#_Toc433738243)

[4.1.4. Herramientas a Utilizar. 53](#_Toc433738251)

[4.1.5. Objetivos del Proyecto. 54](#_Toc433738252)

[4.1.6. Análisis y gestión del riesgo. 55](#_Toc433738253)

[4.1.6.1. Desarrollo del análisis de Riesgos. 57](#_Toc433738254)

[4.1.6.2. Línea de Corte. 58](#_Toc433738255)

[4.1.7. Planificación Inicial. 63](#_Toc433738256)

[4.1.8. Estudio de Factibilidad. 69](#_Toc433738257)

[4.1.8.1. Factibilidad Técnica. 69](#_Toc433738258)

[4.1.8.2. Factibilidad Operativa. 69](#_Toc433738259)

[4.1.8.3. Factibilidad Legal. 70](#_Toc433738260)

[4.1.8.4. Factibilidad Económica. 70](#_Toc433738261)

[4.2. Planificación 71](#_Toc433738262)

[4.2.1. Definición y Descripción de Requerimientos. 71](#_Toc433738263)

[4.2.1.1. Interfaces externas 74](#_Toc433738264)

[4.2.1.2. Interfaces de Software. 75](#_Toc433738265)

[4.2.1.3. Interfaz de comunicación. 76](#_Toc433738266)

[4.2.1.4. Requerimientos Funcionales SICOBO. 76](#_Toc433738267)

[4.2.1.5. Requerimientos No Funcionales. 81](#_Toc433738268)

[4.2.1.6. Actores. 82](#_Toc433738269)

[4.2.1.7. Casos De Uso. 82](#_Toc433738270)

[4.2.1.8 Arquitectura apropiada para la solución. 90](#_Toc433738282)

[4.2.2 Diseño Lógico. 91](#_Toc433738283)

[4.2.2.1 Diagramas de Secuencia. 91](#_Toc433738284)

[4.2.2.2 Diagrama de Clases. 94](#_Toc433738285)

[4.2.2.3 Diseño de interfaces de usuario. 96](#_Toc433738286)

[4.2.3 Diseño Físico. 99](#_Toc433738287)

[4.2.3.1 Diagrama de Implementación. 99](#_Toc433738288)

[4.2.3.2 Modelo Físico de Base de Datos 99](#_Toc433738289)

[100](#_Toc433738290)

[4.3 Desarrollo 101](#_Toc433738291)

[4.3.1. Estándar de diseño de base de datos. 101](#_Toc433738293)

[4.3.1.1. Nombre de los objetos de una base de datos. 101](#_Toc433738294)

[4.3.1.2. Sentencias SQL. 101](#_Toc433738295)

[4.3.2. Estándar de Programación de Aplicaciones. 102](#_Toc433738296)

[4.3.2.1. Extensiones de archivos. 102](#_Toc433738297)

[4.3.2.2. Convenciones De Nombres. 102](#_Toc433738298)

[4.3.3 Api REST. 103](#_Toc433738301)

[4.3.3.1 Capa de datos 104](#_Toc433738302)

[4.3.3.2 ORM - Django. 105](#_Toc433738303)

[4.3.3.3 Capa de negocios. 105](#_Toc433738304)

[4.3.4 SICOBO - AngularJS. 106](#_Toc433738305)

[4.3.4.1 Capa de servicios 106](#_Toc433738306)

[4.3.4.2 Capa de controladores 107](#_Toc433738307)

[4.3.4.3 Capa de Vistas. 107](#_Toc433738308)

[4.4 Estabilización 108](#_Toc433738309)

[4.4.1 Revisión del Sistema. 108](#_Toc433738310)

[4.4.2 Plan de Pruebas. 115](#_Toc433738311)

[4.5 Instalación 118](#_Toc433738312)

[4.5.1 Plan de Instalación y Soporte. 118](#_Toc433738313)

[CONCLUSIONES 128](#_Toc433738314)

[RECOMENDACIONES 129](#_Toc433738315)

**GLOSARIO**

**BIBLIOGRAFÍA**

**ANEXOS**

ÍNDICE DE TABLAS

[**Tabla 1-1:** Conceptos de AngularJS 19](#_Toc433639487)

[**Tabla 2-1:** Directivas de AngularJS 22](#_Toc433639488)

[**Tabla 3-2:** Definición de Indicadores. 35](#_Toc433639489)

[**Tabla 4-3:** Características de Hardware 39](#_Toc433639490)

[**Tabla 5-3:**Resultados de CPU en % 39](#_Toc433639491)

[**Tabla 6-3:** Resultados de CPU expresados en GHz 40](#_Toc433639492)

[**Tabla 7-3:** Resultados de CPU expresados en Hz 40](#_Toc433639493)

[**Tabla 8-3:**Resultados Memoria RAM 41](#_Toc433639494)

[**Tabla 9-3:**Resultados Disco 41](#_Toc433639495)

[**Tabla 10-3:**Resultados Red 42](#_Toc433639496)

[**Tabla 11-3:**Resultados Tarjeta Gráfica 43](#_Toc433639497)

[**Tabla 12-3:** Resultados Tarjeta Gráfica Utilizada por cada Framework. 43](#_Toc433639498)

[**Tabla 13-3:** Resultados por indicador y método de cada Framework 44](#_Toc433639499)

[**Tabla 14-3:** Resultados y Porcentajes. 45](#_Toc433639500)

[**Tabla 15-3:** Porcentajes de Consumo o Utilización de Recursos. 47](#_Toc433639501)

[**Tabla 16-3:** Porcentajes de Consumo de Recursos, por ponderación de indicador. 47](#_Toc433639502)

[**Tabla 17-3:** Porcentaje Final, Consumo o Utilización de Recursos. 48](#_Toc433639503)

[**Tabla 18-3:** Porcentajes de Rendimiento 48](#_Toc433639504)

[**Tabla 19-4:** Herramientas a Utilizar 52](#_Toc433639505)

[**Tabla 20-4:** Riesgos 54](#_Toc433639506)

[**Tabla 21-4:** Determinación de Impacto 55](#_Toc433639507)

[**Tabla 22-4:** Probabilidad de que el riesgo ocurra. 55](#_Toc433639508)

[**Tabla 23-4:** Determinación de exposición al riesgo. 55](#_Toc433639509)

[**Tabla 24-4:** Código de colores 56](#_Toc433639510)

[**Tabla 25-4:** Análisis de Riesgo 56](#_Toc433639511)

[**Tabla 26-4:** Línea de Corte 57](#_Toc433639512)

[**Tabla 27-4:** Riesgo 5 58](#_Toc433639513)

[**Tabla 28-4:** Riesgo 6 59](#_Toc433639514)

[**Tabla 29-4:** Riesgo 7 60](#_Toc433639515)

[**Tabla 30-4:** Riesgo 3 61](#_Toc433639516)

[**Tabla 31-4:** Equipo de desarrollo y Gestión 62](#_Toc433639517)

[**Tabla 32-4:** Costo Proyecto 69](#_Toc433639518)

[**Tabla 33-4:** Requisitos Servidor 73](#_Toc433639519)

[**Tabla 34-4:** Requisitos Cliente 73](#_Toc433639520)

[**Tabla 35-4:** Interfaces de Software 74](#_Toc433639521)

[**Tabla 36-4:** Caso de Uso. Requerimiento # 1 81](#_Toc433639522)

[**Tabla 37-4:** Caso de Uso. Requerimiento # 2 83](#_Toc433639523)

[**Tabla 38-4:** Caso de Uso. Requerimiento # 3 84](#_Toc433639524)

[**Tabla 39-4:** Caso de Uso. Requerimiento # 4 85](#_Toc433639525)

[**Tabla 40-4:** Caso de Uso. Requerimiento # 5 87](#_Toc433639526)

[**Tabla 41-4:** Caso de Uso. Requerimiento # 6 88](#_Toc433639527)

[**Tabla 42-4:** Archivos con sus extensiones 101](#_Toc433639528)

[**Tabla 43-4:** Módulos A Evaluar 115](#_Toc433639529)

ÍNDICE DE FIGURAS

[**Figura 1-1.**  Los servicios Web en Funcionamiento 8](#_Toc433639530)

[**Figura 2-1.** Servicio Web REST. 10](#_Toc433639531)

[**Figura 3-1.** Arquitectura MVC. 14](#_Toc433639532)

[**Figura 4-1.** Instalación AngularJS. 17](#_Toc433639533)

[**Figura 5-1.**  Ejemplo Framework AngularJS 18](#_Toc433639534)

[**Figura 6-1.**  Two-Way Data Binding. 21](#_Toc433639535)

[**Figura 7-1.** Scope en AngularJS. 26](#_Toc433639536)

[**Figura 8-1.**  Resultado App con EmberJS 32](#_Toc433639537)

[**Figura 9-2.** Prototipo del Framework AngularJs 37](#_Toc433639538)

[**Figura 10-2.**  Datos ingresados en el Prototipo AngularJs 37](#_Toc433639539)

[**Figura 11-2.** Prototipo del Framework EmberJS 38](#_Toc433639540)

[**Figura 12-2.** Datos ingresados en el Prototipo EmberJS 38](#_Toc433639541)

[**Figura 13-3.** Resultados, Consumo de Recursos. 46](#_Toc433639542)

[**Figura 14-3.** Resultados de Rendimiento. 48](#_Toc433639543)

[**Figura 15-4.** Orgánico Estructural Zonal. 64](#_Toc433639544)

[**Figura 16-4.** Orgánico Dirección Provincial. 64](#_Toc433639545)

[**Figura 17-4.**  Diagrama de Gantt 66](#_Toc433639546)

[**Figura 18-4.** Diagrama de Gantt en Seguimiento 67](#_Toc433639547)

[**Figura 19-4.** Caso de Uso. Requerimiento # 1 81](#_Toc433639548)

[**Figura 20-4.** Caso de Uso. Requerimiento # 2 82](#_Toc433639549)

[**Figura 21-4.** Caso de Uso. Requerimiento # 3 84](#_Toc433639550)

[**Figura 22-4.** Caso de Uso. Requerimiento # 4 85](#_Toc433639551)

[**Figura 23-4.** Caso de Uso. Requerimiento # 5 86](#_Toc433639552)

[**Figura 24-4.** Caso de Uso. Requerimiento # 6 88](#_Toc433639553)

[**Figura 25-4.** Arquitectura Interna 89](#_Toc433639554)

[**Figura 26-4.** Arquitectura Externa 90](#_Toc433639555)

[**Figura 27-4.** Req.1 Diagrama de secuencia de Administración de asociaciones 90](#_Toc433639556)

[**Figura 28-4.** Req.2 Diagrama de secuencia de Registro de ventas. 91](#_Toc433639557)

[**Figura 29-4.** Req.3 Diagrama de secuencia de Registro de compras 91](#_Toc433639558)

[**Figura 30-4.** Req.4 Diagrama de Secuencia de Administración de inventario 92](#_Toc433639559)

[**Figura 31-4.** Req.5 Diagrama de secuencia de gestión de productos caducados 92](#_Toc433639560)

[**Figura 32-4.** Req.6 Diagrama de Secuencia de generación de reportes. 93](#_Toc433639561)

[**Figura 33-4.** Diagrama de Clases. 94](#_Toc433639562)

[**Figura 34-4.** Registro de ventas 95](#_Toc433639563)

[**Figura 35-4.** Registro de ventas 95](#_Toc433639564)

[**Figura 36-4.** Registro de compras 96](#_Toc433639565)

[**Figura 37-4.** Registro de compras 96](#_Toc433639566)

[**Figura 38-4.** Inventarios 97](#_Toc433639567)

[**Figura 39-4.** Inventarios 97](#_Toc433639568)

[**Figura 40-4.** Diagrama de Implementación 98](#_Toc433639569)

[**Figura 41-4.** Modelo Físico BD 99](#_Toc433639570)

[**Figura 42-4.** Tablas de la base de datos 103](#_Toc433639571)

[**Figura 43-4.** Negocios – módulo de ventas 104](#_Toc433639572)

[**Figura 44-4.** Capa de servicios 105](#_Toc433639573)

[**Figura 45-4.** Capa de controladores 106](#_Toc433639574)

[**Figura 46-4.** Capa de Vistas 107](#_Toc433639575)

[**Figura 47-4.** AngularJS GET, CPU y Memoria RAM 134](#_Toc433639576)

[**Figura 48-4.** EmberJS GET, CPU y Memoria RAM 134](#_Toc433639577)

[**Figura 49-4.** AngularJS POST, CPU y Memoria RAM 135](#_Toc433639578)

[**Figura 50-4.**  EmberJS POST, CPU y Memoria RAM 135](#_Toc433639579)

[**Figura 51-4.** AngularJS GET, Disco 136](#_Toc433639580)

[**Figura 52-4.** EmberJS GET, Disco 136](#_Toc433639581)

[**Figura 53-4.** AngularJS POST, Disco 137](#_Toc433639582)

[**Figura 54-4.** EmberJS POST, Disco 137](#_Toc433639583)

[**Figura 55-4.** AngularJS GET, Red 138](#_Toc433639584)

[**Figura 56-4.** EmberJS GET, Red 138](#_Toc433639585)

[**Figura 57-4.** AngularJS POST, Red 139](#_Toc433639586)

[**Figura 58-4.** EmberJS POST, Red 139](#_Toc433639587)

[**Figura 59-4.** AngularJS GET, Tarjeta Gráfica 140](#_Toc433639588)

[**Figura 60-4.** EmberJS GET, Tarjeta Gráfica 140](#_Toc433639589)

[**Figura 61-4.**  Tarjeta Gráfica sin ejecución de prototipos 141](#_Toc433639590)

ÍNDICE DE ANEXOS

[**Anexo A:** Resultados, pruebas con los Frameworks, AngularJS y EmberJS](#_Toc433617213)

[**Anexo B:** Utilización de Tarjeta Gráfica por el computador, sin ejecución de los prototipos.](#_Toc433617214)

# RESUMEN

Se analizó frameworks JavaScript MVC (Modelo, Vista, Controlador), del lado del cliente para el desarrollo de aplicaciones web. Para el desarrollo de esta investigación se tomaron dos de los frameworks JavaScript MVC (Modelo, Vista, Controlador) más utilizados actualmente, AngularJs y EmberJs. Los mismos que se definen con un propósito, funcionalidad y características semejantes. Se utilizaron técnicas de, observación, entrevistas y estadística descriptiva con la finalidad de recopilar información, analizar e interpretar los resultados para el análisis comparativo de los frameworks, especificando como factor de comparación el rendimiento en cuanto a consumo de hardware en el lado del cliente. Para la demostración de la hipótesis se tomó los indicadores de rendimiento en lo que se refiere a consumo de hardware los siguientes: Tarjeta Gráfica, Memoria RAM, CPU, Disco Duro, Red, en donde se comprobó que la utilización del framework JavaScript MVC AngularJs, cumple con los requisitos alcanzando 48.89% de rendimiento. Analizando y comparando los frameworks JavaScript MVC más destacados para el desarrollo de aplicaciones web: AngularJs y EmberJs, se eligió AngularJs como el más adecuado, porque alcanzó el mayor puntaje 48.89% de rendimiento, en comparación con 30.83% obtenido por EmberJs, lográndose el objetivo de esta investigación. Por lo tanto, el framework de mayor rendimiento es AngularJs puesto que, en relación con el frameworkEmberJs existe una diferencia de 18.06%, por lo que se lo seleccionó para el desarrollo de la aplicación web denominada: SICOBO (Sistema de Control de Botiquín Veterinario para el Ministerio de Agricultura, Ganadería, Acuacultura y Pesca de Morona Santiago. Para la realización de un análisis comparativo entre Frameworks de desarrollo Web, es necesario elegir cuidadosamente a los involucrados, los cuales deben tener funcionalidad y características semejantes para obtener resultados certeros.

**Palabras clave:** <ANÁLISIS COMPARATIVO>, < HERRAMIENTA DE PROGRAMACIÓN [FRAMEWORK ANGULARJS] >, < [DESARROLLO DE APLICACIONES WEB]>, <BOTIQUÍN VETERINARIO>, < [MAGAP] MINISTERIO DE AGRICULTURA, GANADERÍA, ACUACULTURA Y PESCA >

# ABSTRACT

Frameworks JavaScript MVC (Model, View, and Controller) was analyzed by customers to develop web applications. AngularJs and EmberJs were taken as framework JavaScript MVC for this research. They are defined with a target, functionality, and similar characteristics. Techniques such as: observation, interview, and descriptive statistics were used in order to collect information, analyze and interpret results for frameworks comparative analysis specifying as comparison factor performance related to hardware consume considering customers view. Performance parameter related to hardware consume for example: Graphics Card, RAM memory, Hard Disc, Network were considered to prove hypothesis. Framework JavaScript MVC AngularJs complies with all requisites reaching 48.89% of performance. After comparing the web applications AngularJs and EmberJs, the first one was chosen because it has the best punctuation 48.89% against 30.83% obtained by the second one, the difference between them is 18.06%, so the research objective was reached. Base on is mention above AngularJs was select to the web application named: SICOBO (Veterinarian First-aid-kit Control System, for the Ministry of Agriculture, Livestock, Aquaculture, and Fish of Morona Santiago province. To carry out the comparative analysis between web developments frameworks it is important to choose involved ones carefully because they must have functionality and similar characteristics to get concrete results.

**Key words:** <COMPARATIVE ANALYSIS>, < PROGRAMMING TOOLS [ANGULARJS FRAMEWORK] >, < [WEB DEVELOPMENT FRAMEWORK]>, <VETERINARIAN FIRST-AID-KIT>, < [MAGAP] MINISTRY OF AGRICULTURE, LIVESTOCK, ACUACULTURE, AND FISH>

# INTRODUCCIÓN

Se han desarrollado gran cantidad de Frameworks que implementan el patrón MVC para el desarrollo de aplicaciones web, entre los cuales se destacan: Ruby on Rails, CodeIgniter, Django, Catalyst, CakePHP. Uno de los retos importantes del diseño es la selección oportuna y adecuada de dichos elementos, en particular porque existen cientos de patrones de diseño y una gran variedad de frameworks.

Actualmente, con un mundo de tecnología tan creciente y las nuevas tendencias de los usuarios, muchos de los sitios web se han desarrollado para el entretenimiento y comunicación, además llegando a ser indispensables para empresas que desean ser muy competitivas para así poder prevalecer en su nicho de mercado.

La gran demanda exige que las aplicaciones sean livianas e independientes de la plataforma en la que se ejecuten, esto ha permitido que lenguajes como JavaScript sean tomados muy en serio por las comunidades y expertos desarrolladores de aplicaciones web, los Frameworks JavaScript MVC AngularJS y EmberJS son el resultado de ello. Al ser código JavaScript, permite que las aplicaciones web se ejecuten en el cliente y no en el servidor, alivianando la carga para este.

AngularJS fue creado, como un proyecto paralelo, en el año 2009 por dos desarrolladores, Misko Hevery y Adam Abrons. Los dos se previeron inicialmente su proyecto, GetAngular, para ser una herramienta de extremo a extremo que permite a los diseñadores web interactuar tanto con el frontend y el backend.

Hevery finalmente comenzó a trabajar en un proyecto en Google llamado Google Feedback. Hevery y otros 2 desarrolladores escribieron 17.000 líneas de código en el período de 6 meses para Google Feedback. Sin embargo, como el tamaño del código aumentó, Hevery comenzó a crecer frustrado con lo difícil que era para probar y modificar el código que el equipo había escrito.

Así Hevery hizo la apuesta con su manager que podía reescribir toda la aplicación utilizando su proyecto GetAngular lado en dos semanas. Hevery perdió la apuesta. En lugar de 2 semanas le tomó 3 semanas para volver a escribir toda la aplicación, pero él fue capaz de cortar la solicitud de 17.000 líneas a 1500 líneas.

Debido al éxito de Hevery su manager se dio cuenta, y las cosas empezaron a acelerar a partir de ahí. Uno de los creadores originales, Adam Abrons dejó de trabajar en AngularJS pero Misko Hevery y su manager, Brad Green, hizo girar el proyecto original GetAngular en un nuevo proyecto, lo nombró AngularJS, y construyó un equipo para crear una mantenerla dentro de Google.

EmberJS Se inició en SproutCore, la historia de lo que puede iluminar el diseño de EmberJS y otros entornos de aplicaciones de una sola página. SproutCore se remonta a 2007, y es mejor conocido como el marco detrás de MobileMe de Apple (ahora iCloud) La suite de aplicaciones web. Fue creado para construir aplicaciones de estilo de escritorio en el navegador y le dio un enfoque basado en widgets de interfaz de usuario.

El creador de SproutCore, Charles Jolley, trabajó en el marco en Apple hasta 2010, cuando dejó de formar Strobe Inc. Se le unieron, entre otros, compañero de Apple ingeniero Tom Dale y prolífico colaborador de código abierto Yehuda Katz, conocido por su trabajo en Rubí on Rails y jQuery.

Como el desarrollo de SproutCore avanzaba, los objetivos del equipo divergieron. SproutCore tenía fijaciones de gran alcance, los observadores y los fundamentos MVC bien diseñados, pero también se centró en gran medida en un diseño de interfaz que imitaba a las aplicaciones de escritorio y carecía de un sistema de plantillas intuitiva. En 2011, una reescritura completa fue anunciada bajo el nombre de "Proyecto Ámbar".

El plan era tomar lo mejor de SproutCore, casarla con plantillas auto-actualización declarativa utilizando la biblioteca Manillar de Yehuda, y poner de HTML y CSS frontal y el centro como la capa de presentación. En Diciembre de 2011, the SproutCore 2.0 Framework fue renombrado a Ember.js.

**Antecedentes**

Cada vez que nuevos programadores se interesan en el desarrollo web mediante JavaScript MVC tienen dificultades al elegir el Framework a utilizar. La interfaz es muy importante cuando a rendimiento y consumo de recursos se trata, en ocasiones se requiere que las páginas web respondan con gran velocidad, en otras que no consuman tantos recursos de hardware del servidor y en otras ocasiones que al momento de hablar de rendimiento sean muy prácticos, en todo esto radica la indecisión de los programadores.

El estado ecuatoriano se encuentra aplicando un cambio gigantesco en cuanto a manejo de la información se refiere, se está pasando de manejar documentación física a manejar documentación digital.

En el MAGAP en su dirección provincial de morona Santiago, se ha encontrado un problema en cuanto a esto, puesto que al momento de brindar información rápida y oportuna acerca de los botiquines veterinarios de cada cantón de la provincia, no se la puede proporcionar forma ágil, envista que este proceso es realizado manualmente, generando pérdidas de tiempo, recursos, dinero y principalmente malestar entre los funcionarios de la institución.

Es por ello que se ha planteado el proyecto de investigación entre estos frameworks de desarrollo web, hemos escogido los dos más utilizados que son: AngularJS y EmberJS, para saber cuál de ellos brinda un mejor rendimiento en una aplicación web en tiempo de ejecución y además desarrollar la aplicación web que permitirá mejorar los tiempos de respuesta en el botiquín veterinario del MAGAP.

**Delimitación**

El análisis comparativo se realizará entre los frameworks AngularJS y EmberJS, para obtener información clara y precisa acerca de cuál de estos permitirá obtener el mejor rendimiento para desarrollar aplicaciones web.

Refiriéndose por **rendimiento** al consumo de recursos hardware en el lado del cliente como son los siguientes indicadores: Consumo de Tarjeta Gráfica, Consumo de Memoria RAM, Consumo del CPU, Consumo del Disco Duro, Consumo de Red.

Los indicadores serán evaluados a través de la construcción de dos prototipos, el primero desarrollado con el framework AngularJS y el segundo desarrollado con el framework EmberJS, cada uno con las mismas funciones básicas y en base a estos resultados se desarrollara la aplicación web a la que llamaremos “SICOBO” (sistema de control de botiquín) para el MAGAP de Morona Santiago.

La aplicación automatizará los procesos que se llevan a cabo manualmente acerca del botiquín que maneja el MAGAP.

**Justificación Teórica**

El poder disminuir la carga de trabajo de una aplicación web del lado del servidor requiere utilizar las mejores herramientas que permitan que esto sea posible, esto se da debido al auge que ha tenido el internet, las aplicaciones de escritorio tienden a reemplazadas por aplicaciones web, para cumplir con el objetivo de acceso a la información en el momento que se requiera de una manera sencilla y práctica.JavaScript (MVC) tiene una arquitectura de software mejorada, que simplifica el desarrollo de aplicaciones web.

Cada vez más es creciente el número de desarrolladores que se inclinan por la utilización de este patrón, optando cada uno por la elección de algún Framework que simplifiquen y se adapten a sus necesidades al momento de desarrollar sistemas web; es por ello que esta investigación pretende ser una herramienta de decisión para todos aquellos que buscan inmiscuirse en este campo.

**Justificación Aplicativa**

El Ministerio de Agricultura, Ganadería, Acuacultura y Pesca es la institución rectora del multi-sector, para regular, normar, facilitar, controlar, y evaluar la gestión de la producción agrícola, ganadera, acuícola y pesquera del país; promoviendo acciones que permitan el desarrollo rural y propicien el crecimiento sostenible de la producción y productividad del sector impulsando al desarrollo de productores, en particular representados por la agricultura familiar campesina, manteniendo el incentivo a las actividades productivas en general.

Con el propósito de cumplir su misión se realizan inspecciones en distintos lugares de trabajo para ejercer la gestión del desarrollo social. Es por esto que en el MAGAP existe el área de desarrollo productivo el mismo que constantemente se encuentra movilizándose teniendo como objetivo el desarrollo del sector.

De la misma forma el MAGAP cumple con las responsabilidades institucionales que son el objetivo de su existencia, pero al momento de registrar e inspeccionar los botiquines veterinarios referidos en cada cantón de la provincia, se lo hace de manera física, manual, por carpetas o archivos.

Existen varios miembros del departamento de Desarrollo Productivo, todos realizan inspecciones, pero al momento de llevar un control de los productos veterinarios existentes de cada Sub Zona cantonal, no cuentan con una base de datos de estos lo cual dificulta potencialmente la planificación eficiente del desarrollo de las actividades.

No se sabe que producto se utilizó y quien lo demando, cuanto producto se encuentra disponible, si se realizaron compras, para quien se realizan las ventas, que productos están caducados o por caducar. Cada dirección provincial tiene gran dificultad al momento de presentar información oportuna y precisa acerca del uso de los medicamentos veterinarios de cada cantón.

Ante esta necesidad, es imperante que dicho departamento cuente con un sistema informático capaz de acceder a esta información, generando reportes que permitan a los miembros del departamento desarrollo social y productivo, planificar eficientemente sus actividades y generar reportes exactos del trabajo realizado, sistema al que llamaremos “SICOBO”.

El área de Desarrollo Social y Productivo del MAGAP en Morona Santiago contará con un sistema de control de botiquín veterinario, que mejorará notablemente la planificación y organización de las inspecciones a los diferentes lugares de trabajo de la provincia, además de automatizar los procesos que se llevan a cabo manualmente acerca del botiquín que se maneja en cada cantón brindando información rápida, ágil y oportuna en el momento que se requiera.

**Objetivo General**

Realizar un análisis comparativo entre los Frameworks MVC de JavaScript AngularJS y EmberJS para el desarrollo de aplicaciones Web.

**Objetivos Específicos**

* Estudiar los Frameworks MVC de JavaScript AngularJS y EmberJS para el desarrollo de aplicaciones Web.
* Establecer los parámetros de evaluación para elaborar la comparativa entre los Frameworks.
* Realizar el análisis comparativo entre los Framework AngularJS y EmberJS, mediante la evaluación de prototipos y definir el Framework de mejor rendimiento.
* Desarrollar el sistema de control de botiquín veterinario para el MAGAP, Morona Santiago.

**Hipótesis**

El desarrollo de aplicaciones web con el Framework JavaScript MVC AngularJS, permitirá obtener mayor rendimiento que con el Framework EmberJS.

# CAPÍTULO I

## MARCO TEÓRICO

### Aplicación Web



#### Concepto

Es una aplicación que se ejecuta en un servidor, al cual se puede tener acceso mediante un navegador en el internet, en la actualidad se ha vuelto común el uso de este tipo de aplicaciones aún sin darnos cuenta como al momento de enviar un correo electrónico, un movimiento bancario, redes sociales,

Es posible migrar una aplicación de escritorio a una aplicación web, según sea la necesidad, ya que existen programas como videojuegos, o de diseño gráfico complejos que necesitan un ordenador con buenas características, para poder ejecutarse y si estos fuesen migrados a aplicaciones web, los tiempos de respuesta serían prolongados. (Moreira, V., 2015, http://es.scribd.com)

#### Características

* Se puede acceder mediante un navegador web
* Basta con tener acceso a internet y se puede acceder desde cualquier parte del mundo.
* Existe una única aplicación en el servidor, al cual acceden miles de usuarios, permitiendo que las actualizaciones se reflejen inmediatamente.
* Las tecnologías con las que se puede desarrollar estas aplicaciones como, JavaScript, flash, Ajax, Java incrementan la calidad de interfaz de usuario.
* Permite que la aplicación sea multiplataforma. (Alegsa, L., 2015, http://www.alegsa.com.ar).

### Servicios Web

#### Definición

Son el conjunto de aplicaciones o tecnologías que tienen como característica la interoperación en la web, intercambian datos para ofrecer servicios como procedimientos remotos, los proveedores los ofrecen y los clientes los consumen.

A continuación se muestra en la figura 2-1 un ejemplo de estructura de los servicios Web.

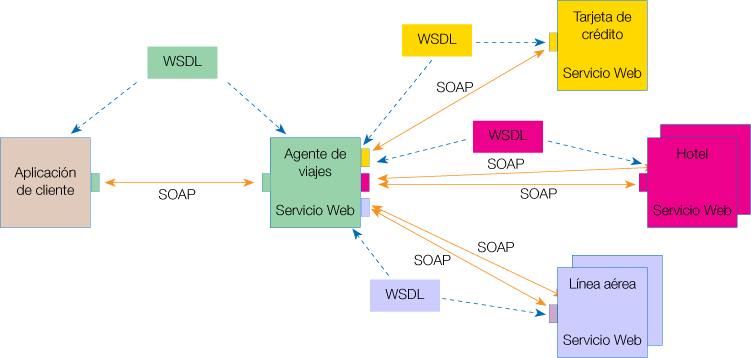


Figura 1-1. Los servicios Web en Funcionamiento

**Fuente:** http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb, 2015.

#### Intervinientes

* Clientes: Realizan las peticiones (navegadores).
* Servidores: Programas que atienden las solicitudes y devuelven una repuesta.
* Aplicaciones de Software: Mecanismo utilizado por los clientes y servidores para poder comunicarse.
* Lenguajes de Programación: Permiten crear las aplicaciones.
* Protocolos: Reglas para la comunicación entre el cliente y el servidor.
* Estándares: Reglas normalizadas para permitir la comunicación entre tecnologías diferentes de hardware y software estableciendo compatibilidad entre ellos. (Mohamed, S., 2015, http://es.scribd.com).

#### Estándares

A continuación una breve descripción de los principales estándares de los servicios web.

* Web Services Protocol Stack, Conjunto de servicios y protocolos de los servicios web.
* XML, Extensible Markup Language, formato de intercambio de datos.
* SOAP, Protocolo de Acceso de Objetos Simples.
* HTTP, Hypertext Transfer Protocol
* FTP, para transferir archivos.
* SMTP, envío de correo electrónico.
* UDDI, para publicar servicios.
* WSDL, Lenguaje de interfaz público para servicios web, basado en XML. (Tavares, J., 2015, http://es.slideshare.net)

### Servicios Web REST

#### Definición de REST

REST*,* Representational State Transfer, es un estilo arquitectónico ligero para desarrollar servicios web, basado principalmente en las URLs a las que se accede mediante HTTP, para obtener recursos o realizar alguna operación, cualquier cosa que se pueda direccionar mediante la web es un recurso REST, la palabra direccionar hace mención a que puede ser accedido y transferido entre cliente y servidor como se muestra en la figura 2-1.

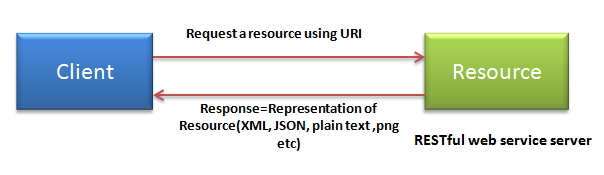


Figura 2-1. Servicio Web REST.

Fuente: http://www.arquitecturajava.com/servicios-rest, 2015.

#### *Características de los Servicios Web REST*

Los servicios REST se caracterizan por seguir las siguientes condiciones.

* Cliente/Servidor: Define una interfaz de comunicación teniendo por separado las responsabilidades entre ambas partes.
* Sin estado: Los servicios no guardan las sesiones de los usuarios, cada petición es independiente de las demás.
* Caché: El contenido de los servicios REST se puede cachear.
* Servicios Uniformes: GET, POST, PUT, DELETE.
* **Arquitectura en Capas: Orientado a ser escalable, el cliente REST no distingue en si la petición la está haciendo al servidor, a un sistema de caches o a un balanceador que redirige a otro servidor. (Álvarez, C., 2015,** http://www.arquitecturajava.com**).**

#### Formato JSON

JavaScript Object Notation- Notación de Objetos de JavaScript, formato ligero de intercambio de datos, aparece como alternativa a XML y puede ser leído por cualquier lenguaje de programación, permitiendo intercambiar información entre diferentes tecnologías.

Está constituido por dos estructuras:

* Una colección de pares de nombre/valor (objeto).
* Una lista ordenada de valores (Arrays).

*Nombre/Valor*

“Nombre”: “Jorge Peña”

*Tipos de valores en JSON*

Número (entero, float)

String (entre comillas)

Booleano (true o false)

Array (entre [])

Objeto (entre {})

Null

*Objetos*

{ "Fruta":"Manzana" , "Cantidad":20 }

*Arrays*

{

"Frutas": [

{ "Fruta":"Papaya" , "cantidad":10 },

{ "Fruta":"Fresa" , "cantidad":20 },

{ "Fruta":"Mandarina" , "cantidad":30 }

]

}. (Esquiva, A., 2015, https://geekytheory.com)

### Tecnología JavaScript

#### Introducción

JavaScript es el lenguaje de programación interpretado más usado actualmente en el desarrollo de aplicaciones Web, contiene una sintaxis muy similar a Java y a C, pero no guarda relación directa con ninguno de estos lenguajes de programación, puesto que no es un lenguaje orientado a objetos propiamente dicho, sino que éste, está basado en prototipos, ósea entre otras palabras, las clases nuevas se generan clonando las clases base y de esta forma se extiende su funcionalidad. (Flanagan, D., 2007, https://books.google.com.ec).

Al ser un lenguaje de programación interpretado no necesita que los programas sean compilados para que se ejecuten ya que el lenguaje funciona del lado del cliente, cualquier código escrito en JavaScript es directamente trasparente y visible por cualquier navegador web, puesto que él es el encargado de interpretar dicho código sin la necesidad de procesos intermedios. (Pérez, D., 2007, http://www.maestrosdelweb.com)

#### ¿Cómo identificar código JavaScript?

El código JavaScript se puede encontrar dentro de las etiquetas <body></body> en las páginas web.

Por lo general se insertan entre: <script></script>.

También pueden estar ubicados en ficheros externos usando:

<script type="text/javascript" src="micodigo.js"></script>

Entre algunos de los servicios que se encuentran realizados con JavaScript en Internet se encuentran:

* Correo
* Chat
* Buscadores de Información
* Sitos de consulta
* Páginas de comunicación
* Tiendas virtuales.

Es decir que se puede encontrar código JavaScript en la gran mayoría de servicios y aplicaciones web publicado en internet o en redes privadas.

#### Ventajas

Entre las ventajas más destacables podemos mencionar las siguientes:

* Útil para el desarrollo de páginas web dinámicas.
* Excelente silicón para la validación de datos de un formulario en el lado del cliente.
* Es un lenguaje sencillo y liviano.
* Consume poca memoria.
* Tiene gran cantidad de efectos visuales.
* Compatible con la gran mayoría de navegadores modernos y fácil manejo de datos.
* Se recomienda para la creación de aplicaciones web.
* El código se ejecuta en el lado del cliente, por lo que el servidor no es solicitado más de lo debido.
* No necesita de un compilador para ejecutarse, sino es el navegador el que interpreta el código.

#### Desventajas

Una de las desventajas más notable de JavaScript es la seguridad, el código es visible y puede ser leído por cualquiera. Es verdad que hoy en día existen varios estándares de seguridad que restringen la ejecución de código por parte de los navegadores, pero aun así se puede ejecutar código malicioso que dañe, robe o destruya información del lado del cliente.

Los <script> tienen capacidades limitadas, por razones de seguridad, por lo cual no es posible hacer todo con JavaScript, sino que es necesario usarlo conjuntamente con otros lenguajes evolucionados, posiblemente más seguros, como Java. Existe un determinado número de etiquetas, por lo que no se pueden crear más.

### Arquitectura MVC

#### Patrón de arquitectura Modelo Vista Controlador (MVC)

El patrón de arquitectura de software MVC (Modelo Vista Controlador) define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (De Flujo de trabajo de la aplicación).

Entre otras palabras la arquitectura MVC separa los datos y la lógica de negocio, de la interfaz de usuario.

Es así entonces que, dividimos el sistema en tres capas en donde se obtiene, la encapsulación de los datos, la interfaz o vista y por último la lógica interna o controlador, la siguiente figura 3-1 detalla la arquitectura MVC.

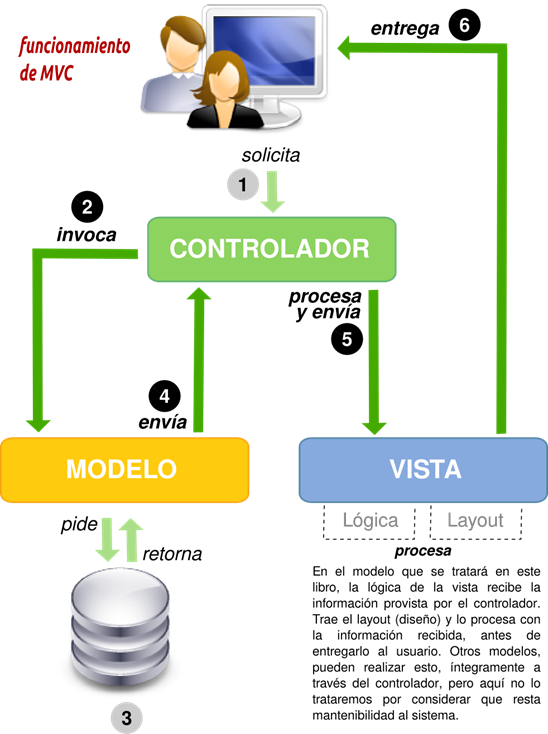


Figura 3-1. Arquitectura MVC.

Fuente: http://librosweb.es/libro/jobeet\_1\_4/capitulo\_4/la\_arquitectura\_mvc.html, 2015.

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones compuesta por tres capas:

#### Capa Modelo

Esta capa del modelo define la lógica de negocio (la base de datos pertenece a esta capa). En esta sección estará la estructura de datos, junto con las clases relacionadas a la base de datos y los métodos respectivos.

#### Capa Vista

La vista es lo que se muestra al usuario para que pueda interactuar con aplicación (A esta capa pertenecen los templates o plantillas). Proporcionará al cliente múltiples páginas web dinámicas, visualizándose para él como simples páginas HTML. Esto permite que personas con conocimientos de HTML puedan desarrollar sus aplicaciones de forma sencilla.

#### Capa Controlador

El controlador es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Este modelo de arquitectura de aplicaciones brinda varias ventajas importantes al desarrollo de aplicaciones web, entre las cuales se puede mencionar las siguientes:

* Dividir la lógica de negocio del diseño, haciendo el proyecto más escalable, no usa el estado de vista, ni formularios basados en servidor. Haciendo que el marco de MVC sea ideal para los desarrolladores que deseen un control completo sobre el comportamiento de la aplicación.
* Los frameworks están creados para facilitar el trabajo de los desarrolladores, encontrarás clases para controlar fechas, URL's, Webservices. lo que tiene una gran ventaja en cuanto a productividad. Inicialmente como es lógico habrá una curva de aprendizaje, pero luego tendrás muchos beneficios.

### Framework AngularJS

#### Introducción

Diseñado para aplicaciones web dinámicas, mantenido por Google, utiliza HTML como lenguaje de plantillas y la extiende para expresar los componentes de su aplicación, luego AngularJS enseña al navegador la nueva sintaxis a lo que llamamos directivas.

Los conceptos de enlace de datos e inyección de dependencias permite reducir líneas de código y su ejecución es realizada dentro del navegador.

Angular es 100% JavaScript del lado del cliente, no es un simple sistema de plantillas, la razón se debe a “bidirectional data binding”, enlace de datos bidireccional, que posee y se ejecuta de forma automática. La plantilla se compila en el navegador y el paso de compilación da como resultado una vista en vivo. Ya no se necesita que los desarrolladores deban sincronizar constantemente la vista con el modelo o viceversa.

Además tampoco es necesario dejar de lado jQueryUI o Bootstrap. Angular trabaja muy bien con librerías de componentes de terceros. Ayuda a la gestión de lo que se conoce como aplicaciones de una sola página. (docs.angularjs.org, 2015, https://docs.angularjs.org).

Angular permite la utilización de cualquier tipo de backend para trabajar con la aplicación, existe libertad en la utilización de Java, Phyton, Ruby, C# o cualquier otro lenguaje. Lo que si se necesita es una forma de comunicación de ida y vuelta con el servidor, peticiones HTTP XML o JSON.

Si el servidor posee un API REST que provee valores JSON, trabaja muy fácilmente con el frontend, aunque se debe decir que no es un problema si el servidor no devuelve JSON, porque se le puede enseñar a AngularJS a comunicarse con el servidor XML o con algún otro formato. (Seshadri, S., Green, B., 2014, pp.3-11).

#### *Instalación*

AngularJS es JavaScript, por lo tanto basta con descargar el archivo e incluirlo al proyecto. La figura 4-1 muestra la interfaz de descarga de AngularJS.

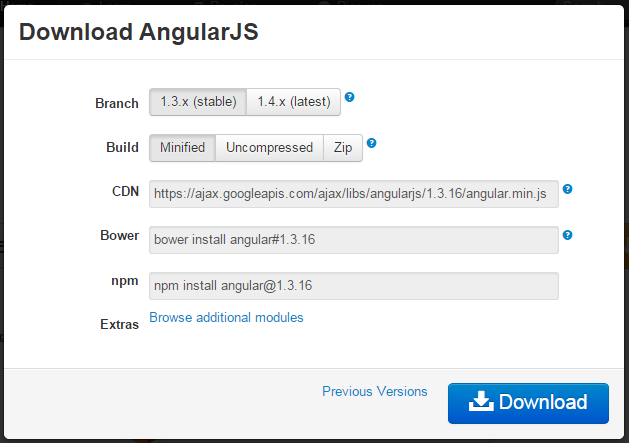


Figura 4-1. Instalación AngularJS.

Fuente: http://www.cursoangularjs.es/doku.php?id=unidades:01\_introduccion:02\_instalacion, 2014.

Cada una de las opciones:

* Branch: Versión para descargar, 1.3 o 1.4
* Build:
* Minified: Archivo angular.js minimizado.
* Uncompressed: Archivo angular.js.
* Zip: Contiene todos los archivos de Angular.
* CDN: Archivo angular.js alojado en el servidor de Google.
* Bower: Gestor de paquetes para la web.
* Extras: Muestra todos los archivos de Angular pero de todas versiones. (González, L., 2014, http://www.cursoangularjs.es)

Este es un ejemplo de aplicación básica con AngularJS Utilizando el CDN de Google.

<!DOCTYPE html>

**<html** ng-app**>**

**<body>**

**<h1>** Hola {{5 + 2}} **</h1>**

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>

**</body>**

**</html>**

La directiva ng- app, marca el inicio de la utilización de AngularJS, al ponerlo junto a la etiqueta HTML permite que AngularJS controle la aplicación desde esta etiqueta, también se puede colocar dentro de cualquier otro elemento, y todo lo que sea hijo de éste también se manejará con AngularJS.

Las expresiones encerradas entre llaves dobles es una sintaxis de AngularJS, si es una variable, la interfaz de usuario se mantiene actualizada con los cambios que se puedan dar, y si se trata de una expresión, AngularJS lo evalúa y mantiene actualizado la interfaz con los cambios de valor de la expresión. (Seshadri, S., Green, B., 2014, pp.11-11).

Si todo está correcto debería ejecutarse como se muestra en la figura 5-1

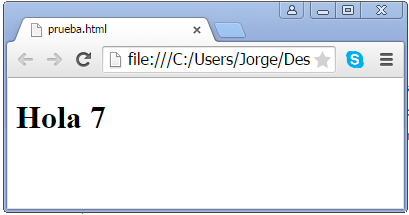
. 

Figura 5-1. Ejemplo Framework AngularJS

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014

#### Conceptos Generales

En la Tabla 1-1 se presenta una breve descripción de los conceptos más importantes de AngularJS. (docs.angularjs.org, 2015, https://docs.angularjs.org).

Tabla 1-1: Conceptos de AngularJS

| **Concepto** | **Descripción** |
| --- | --- |
| Template | HTML con marcado adicional |
| Directives | Extender HTML con atributos y elementos personalizados |
| Model | Datos que se muestran al usuario en la vista |
| Scope | En este se almacena el modelo para que los controladores, directivas y expresiones puedan acceder a él |
| Expressions | Variables de acceso y funciones del ámbito |
| Compiler | analiza la plantilla y crea una instancia, directivas y expresiones |
| Filter | formatea el valor de una expresión para la visualización al usuario |
| View | lo que ve el usuario (DOM) |
| Data Binding | datos de sincronización entre el modelo y la vista |
| Controller | la lógica de negocio detrás de las vistas |
| Dependency Injection | Patrón de diseño de como los componentes se apoderan de sus dependencias |
| Injector | Contenedor de inyección de dependencias |
| Module | Contenedor para las diferentes partes de una aplicación, controladores, servicios, filtros, directivas que configura el inyector |
| Service | lógica de negocio reutilizable independiente de vistas |

**Fuente:** https://docs.angularjs.org/guide/concepts, 2015.

#### Características de AngularJS

##### *Data Binding*

“Enlace de Datos”, Sin duda uno de los puntos fuertes de AngularJS.

El binding es enlazar los datos del Scope con lo que mostramos en el HTML de forma automática, producida en dos sentidos.

* *One-way binding:*

Los datos van únicamente desde el Scope hacia la parte visual, (modelo hacia la vista). Se consigue mediante la sintaxis de llaves dobles.

{{Datos}}

Si se modifica el dato del modelo (Scope), se actualiza automáticamente la vista.

* *Two-way binding:*

Los datos fluyen no solo desde el Scope hacia la vista, sino también desde la vista hacia el Scope, se implementa mediante la directiva ngModel.

<input type="text" ng-model="miDato" />

Cuando el modelo cambia, el dato que está dentro de la caja de texto cambiará y cuando el usuario cambie el valor de la caja de texto el Scope se actualiza automáticamente. La figura 6-1 presenta el funcionamiento del enlace de datos bidireccional de AngularJS. (Álvarez, C., 2014, http://www.arquitecturajava.com).

<div ng-app>

<input type="text" ng-model="dato" /> <!--Doble binding -->

{{dato}} <!--Simple binding -->

</div>

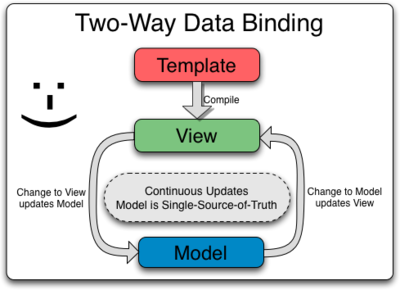


Figura 6-1. Two-Way Data Binding.

**Fuente:** https://docs.angularjs.org/guide/databinding, 2015.

La plantilla (HTML sin compilar, junto con las directivas adicionales), se compila en el navegador. Al realizarse el paso de compilación genera una vista en vivo. Todos los cambios dados en la vista se reflejan al instante en el modelo (Scope) y viceversa.

El término “compilar” una plantilla HTML, para Angular significa agregar los detectores de eventos al HTML. En otras palabras es agregar listeners (escuchadores) de eventos al HTML.

##### *Directivas*

Son marcadores en un elemento DOM (atributo, comentario o clase CSS) estos dan aviso al compilador HTML de angular ($compile), para que sea adjuntado el comportamiento especificado para ese elemento DOM o hasta se podría transformar el elemento DOM y sus hijos. Es la manera en la que se puede extender el HTML, permitiéndole agregar código.

Tipos de Directivas

Las directivas pueden ser basadas en nombres de elementos, atributos, nombres de clase, así como los comentarios, se pueden dar en el nombre de atributo, nombre de la etiqueta, comentarios, o el nombre de la clase de atributos.

<my-dir></my-dir>

<span my-dir="exp"></span>

<!-- directive: my-dir exp -->

<span class="my-dir: exp;"></span>

Aunque puede ser utilizado de distintas maneras es preferible utilizar las directivas en el nombre de la etiqueta y atributos. (docs.angularjs.org, 2015, https://docs.angularjs.org).

Directivas de AngularJS.

La Tabla 2-1 presenta una breve descripción de las directivas nativas de AngularJS, hay que decir que los nombres de las directivas se crean en camelcase (ngApp), pero se separa por un guión (ng-app) en la vista.

Tabla 2-1: Directivas de AngularJS

| **Directiva** | **Descripción** |
| --- | --- |
| *ngApp*  *(ng-app)* | Permite auto arrancar una aplicación Angular, indica el elemento raíz, se coloca como atributo en la etiqueta que quieres que sea la raíz de la aplicación. |
| *ngController*  *(ng-controller)* | Permite indicarle a la vista donde trabajará nuestro controlador y enlazar un $scope, todo modelo que este dentro del ámbito de la directiva podrá ser accedido desde el controlador asignado. |
| *ngModel*  *(ng-model)* | Representa el modelo o dato, permite obtener la información ingresada por el usuario en algún elemento del formulario, basta con asociarle un modelo y éste podrá ser accedido tanto en el controlador como la vista mediante el nombre del modelo. |
| *ngClick*  *(ng-click)* | Trabaja relacionado al evento click, se le puede asociar alguna funcionalidad en cuanto el usuario haga click sobre algún elemento. |
| *ngInit (ng-init)* | Permite evaluar una expresión en el scope donde se está trabajando. |
| *ngRepeat*  *(ng-repeat)* | Permite iterar una colección de datos, generar un template por cada elemento de la colección y pintarlo en la vista, cada template o plantilla recibe su propio ámbito ($scope). |
| Continúa | |
| *ngChange*  *(ng-change)* | Detecta los cambios que se produzcan dentro de una etiqueta de entrada, inputs, checkbox. |
| *ngShow*  *(ng-show) | ngHide*  *(ng-hide)* | Permiten mostrar y ocultar alguna parte de la vista. |
| *ngBind*  *(ng-bind)* | Cumple la misma funcionalidad que las llaves **{{}}**, sin embargo, **ng-bind** tiene una mejor performance en cuanto a tiempo. |

**Fuente:** https://docs.angularjs.org/guide/directive, 2015.

Angular permite crear directivas personalizadas, esta es una forma simpe de implementarla.

var app = angular.module('MiModulo',[]);

app.controller('MiControlador', function($scope){

$scope.cliente = {

nombre: 'Jhon',

direccion: 'Av. Jose pardo 481'

};

});

app.directive('miCliente', function() {

return {

templateUrl: 'cliente.html'

};

});

En la vista

<body>

<div ng-controller="MiControlador">

<div mi-cliente></div>

</div>

</body>

En la vista cliente.html. (Natividad, L., 2015, http://frontendlabs.io)

Nombre: {{cliente.nombre}} Dirección: {{cliente.direccion}}

##### *Expresiones*

Se utilizan para representar datos dentro cualquier parte de un documento HTML, estos pueden ser cadenas de texto números, operaciones matemáticas, variables o funciones, su sintaxis es: {{Dato}}.

Ejemplo, expresiones válidas en angular.(Ruiz, F., 2015, http://fr2dev.com)

Expresiones numéricas

<h1> Cantidad {{7 \* 7}}</h1>

Expresiones con cadenas de texto

<h1> Bienvenido {{“Jorge ” + ”Luis”}}</h1>

Expresiones con Arrays

<h1> Apellido:{{personas[1]}}</h1>

Expresiones con Objetos

<h1> Nombre:{{persona.nombre}}</h1>

##### *Módulos*

Vienen a ser contenedores de distintas partes de la aplicación. Se puede crear la cantidad de módulos que se vaya a utilizar, la idea es desacoplar el código y tratar de agruparlo ya sea por características, por funcionalidades, por componentes reusables. El desacoplar el código facilita mantenerlo y escalarlo.

La forma de declarar un módulo.

angular.module('Nombre\_modulo',[]);

Se utiliza el método module que es nativo de angular, se asigna el nombre que lo identificará y luego se declara las dependencias dentro de los corchetes, en el caso de que se necesite utilizar múltiples dependencias basta con separarlos por comas.

angular.module('Nombre\_modulo',[‘dependencia1’,’dependencia2’,’dependencia3’]);

Se ha declarado el modulo en la parte de javascript, para que funcione hay que enlazarlo a la vista y para ello hay que usar la directiva ng-app.

<!DOCTYPE html>

<html lang="es" ng-app="Nombre\_modulo">

<body>

<h1>AngularJS</h1>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js">

</script>

<script src="app.js">

</script>

</body>

</html>

Se añade la directiva **ng-app** en la etiqueta html y le indicamos el módulo que trabajará en la vista (Nombre\_modulo), se importa el archivo angular.min.js desde el repositorio de google y añadimos el archivo donde se tiene guardado el módulo, en este caso es app.js.

##### *$Scope*

Ámbito o Alcance, es uno de los puntos fundamentales de AngularJS, dentro de este alcance se tiene acceso a las variables y los métodos que enlazan al controlador con la vista. Es decir, todo controlador debe poseer un $scope que le permita interactuar con la vista, en la estructura Model-View-Controller, este objeto se convierte en el modelo, en la figura 7-1 se ilustra la interacción entre sus componentes. (Natividad, L., 2014, http://frontendlabs.io).

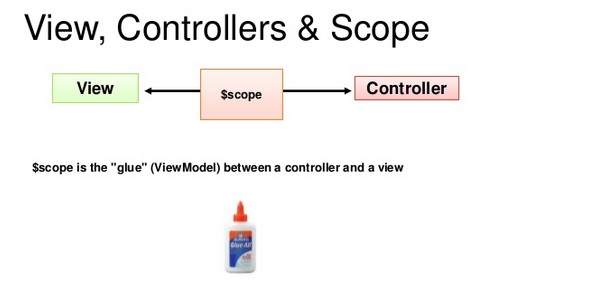


Figura 7-1. Scope en AngularJS.

**Fuente:** http://www.c-sharpcorner.com/UploadFile/dev4634/scope-in-angular-js-for-begineers/, 2015.

##### *Controladores*

De manera general es el encargado de gestionar los eventos, este objeto permite desarrollar la lógica de la aplicación, permite el enlace entre el **scope con la vista, permitiendo tener control total de los datos.** (Natividad, L., 2015, http://frontendlabs.io)

Como declarar un controlador.

var app = angular.module('MyApp', []);

app.controller('nombreControlador', function($scope){

$scope.nombre = "Jorge";

});

Se da un nombre e inyecta las dependencias, todo controlador tiene un $scope asociado, se puede inyectar otras dependencias nativas de AngularJS o personalizados, si se desea inyectar múltiples dependencias hay que separarlos por comas.

var app = angular.module('MyApp', []);

app.controller('nombreControlador', function($scope,dependencia1,dependencia2){

$scope.nombre = "Jorge";

});

Los controladores se enlazan con la vista a través de la directiva ng-controller.

<body>

<div ng-controller="nombreControlador">

<h1>AngularJS{{nombre}}</h1>

<div/>

</body>

##### *Servicios*

Objeto que permite obtener información, un servicio no interactúa con la propia página, sino con otros servicios, con un servidor que pueda estar en otro host. Se caracteriza principalmente porque solo hay una única instancia, pero puede ser utilizado más de una vez, en otras palabras es singleton.

Existen 2 tipos principales de Servicios.

* Service: Se le pasa una clase, AngularJS crea internamente una instancia de la clase, todo lo que se establece en la instancia de nuestro servicio utilizando "this" es público y estará expuesto a cualquier otro controlador o la Directiva donde inyectemos este servicio.
* Factory: Se le pasa una función para que ésta retorne el valor del servicio, devuelve selectivamente lo que queremos exponer. Esto permite tener variables privadas que no puede ser accedidas desde el interior de un controlador. (González, L., 2014, http://www.cursoangularjs.es).

### Framework EmberJS

#### Definición EmberJS

Framework JavaScript de código abierto, está basado en el patrón de arquitectura MVC (modelo, vista, controlador) y orientado a la creación de aplicaciones web dinámicas en el lado del cliente.

Ember está calificado como uno de los principales Framework de JavaScript a nivel mundial, puesto que permite a los diseñadores y desarrolladores crear aplicaciones *single-page* (una sola página).

#### Características generales de uso

* *Configuración* - Usa reglas estrictas para la creación de una potente y sencilla aplicación web, para esto implementa una serie de clases y procedimientos los que nos ayudan a controlar la data, y estandarizar el desarrollo de las aplicaciones del lado del cliente.
* *Enlace de datos* – Para presentar data en las vistas de manera práctica y sencilla Ember.js implementa la librería Handlebars.js, el cual es una dependencia que nos permite definir plantillas o templates semánticas. La características principal es la automatización de cambios, esto quiere decir que, cuando un objeto JavaScript que tengamos enlazado cambia, el DOM automáticamente se actualiza en la app y viceversa, cuando el DOM cambia automáticamente se actualiza el objeto de JavaScript.
* *Mayor enfoque en la creación* - Ember.js es el que se encarga de la parte técnica de la aplicación, permitiendo mayor enfoque en el desarrollo de la misma.

#### Implementar EmberJS

EmberJs implementa tres dependencias fundamentales que son: jquery.js, handlebars.js y ember.js, para crear una aplicación que funcione bajo su código. Estas dependencias se encuentran disponibles para su libre descarga en cada uno de sus portales web.

* La primera dependencia es jQuery “jquery.js”, se puede descargar del sitio: <http://jquery.com/download/>
* La segunda dependencia es Handlebars “handlebars.js”, y se puede descargar del sitio: <http://handlebarsjs.com/>
* La tercera dependencia es el Ember “ember.js”, que también se puede descargar del sitio: <http://emberjs.com/>

Nota: Es importante que las dependencias descargadas sean de la versión de desarrollo puesto que esta nos permitirá conocer claramente los errores presentados en la aplicación en caso que existiera alguno.

#### Mi primera Aplicación con EmberJS

1. Crear el archivo index.html, para ello se genera el siguiente código.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title> Primer App con Ember.js </title>

</head>

<body>

</body>

</html>

1. Se agrega las dependencias descargadas anteriormente.

<script src="jquery.min.js"></script>

<script src="handlebars.js"></script>

<script src="ember.js"></script>

Es importante que la aplicación que se desarrollará, contenga todos los archivos en el mismo nivel, para que no se presente inconvenientes al ejecutar el código, de no cumplirse esto, es necesario especificar la ruta donde se encuentra los recursos, más el nombre del archivo requerido.

Ahora el archivo quedará de la siguiente manera:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Primera App con Ember.js</title>

<script src="jquery.min.js"></script>

<script src="handlebars.js"></script>

<script src="ember.js"></script>

</head>

<body>

</body>

</html>

El estándar de uso de EmberJS es iniciar el App en otro archivo, pero como ejemplo explicativo se realizará dentro del mismo archivo. Con esta nueva línea se crea la aplicación en EmberJS:

<script>

window.App = Ember.Application.create();

</script>

Entonces el archivo index se presentará de esta forma:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title> Primera App con Ember.js </title>

<script src="jquery.min.js"></script>

<script src="handlebars.js"></script>

<script src="ember.js"></script>

<script>

window.App = Ember.Application.create();

</script>

</head>

<body>

</body>

</html>

1. Crear una Plantilla.

Para mostrar información se debe usar a lo que se denomina como Templates o plantilla. Entonces añadir este código.

<script type="text/x-handlebars" data-template-name="index">

<h1>

Bienvenido a Ember.js!

</h1>

</script>

Verificar que el type del script es text/x-handlebars puesto este tag permitirá a Ember.js determinar que esta es una plantilla y que debe ser tratada como tal. Es necesario asignar un nombre a la plantilla *(data-template-name)*, para este ejemplo será index.

Cabe indicar que Ember.js busca los nombres de las plantillas de manera automática, y el index es el primero que trata de mostrar. Entonces el archivo quedaría de la siguiente manera:

<!DOCTYPE html>

<html>

<head>

<title> Primera App con Ember.js </title>

<script src="jquery.min.js"></script>

<script src="handlebars.js"></script>

<script src="ember.js"></script>

<script>

window.App = Ember.Application.create();

</script>

</head>

<body>

<script type="text/x-handlebars" data-template-name="index">

<h1>

Bienvenido a Ember.js!

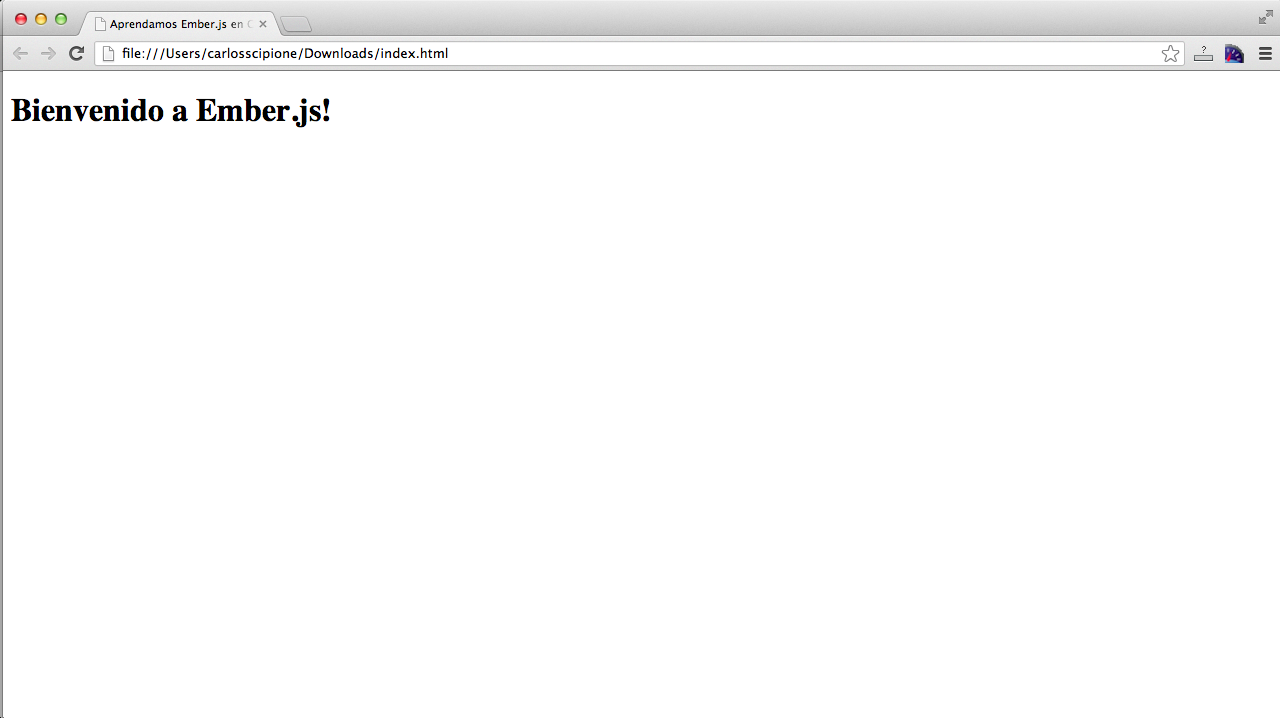
</h1>

</script>

</body>

</html>

Realizado todo el código anterior como resultado se puede observar lo que muestra en la figura 8-1.



**Figura 8-1.** Resultado App con EmberJS

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014

#### Estructura básica de una aplicación con EmberJS

Esta la organización necesaria que debería tener el proyecto, para así entender que es lo que contienen estas principales carpetas.

* **/css** – Contiene los archivos *CSS* que la aplicación necesita para presentar el look deseado por nosotros.
* **/js** – Contiene todos los archivos de JavaScript que nuestra app va a necesitar para poder funcionar y en los cuales se desarrolla el comportamiento de la misma.
* **/app** – Contiene los controladores, modelos, vistas y configuraciones de la aplicación. En esta carpeta escribiremos la mayoría del código para que nuestra aplicación funcione.
* **/app/controllers** – Contiene todos los archivos con las clases de los controladores que sirven para interactuar con los modelos, las vistas y manejar el comportamiento de la aplicación.
* **/app/models** – Los modelos son clases que representan la información (datos) de la aplicación y las reglas para manipularlos.
* **/app/views** – Este directorio contiene los archivos que poseen el comportamiento de renderización de todos los Templates de la aplicación.
* **/app/app.js** – En este archivo van a estar todos las configuraciones necesarias para que nuestro site funcione correctamente.
* **/app/router.js** – En este archivo van a estar definidas todas las rutas que nuestra app va a necesitar.
* **/libs** – En esta carpeta se alojan todas las librerías que conforman el framework y sus dependencias.
* **index.html** – En este archivo van a estar todos los templates necesarios para poder crear el *“workflow”* de la aplicación.

# CAPITULO II

## MARCO METODOLÓGICO

### Introducción

JavaScript uno de los lenguajes más utilizados en la actualidad, debido a que es un lenguaje interpretado, esto hace que el código no necesite ser compilado, sino simplemente puede ser ejecutado en un navegador sin la necesidad de procesos intermedios.

La utilización de JavaScript, potencia las páginas haciéndolas dinámicas, en otras palabras extiende HTML para realizar funciones que no se podían realizar comúnmente, es por ello que se han creado varios Frameworks JavaScript para la ayuda del desarrollo de aplicaciones web y de esta manera facilitar su utilización.

En el presente capítulo se realizó el análisis comparativo acerca del rendimiento de los Frameworks, AngularJs y EmberJS, permitiendo obtener una solución a nuestra aplicación.

Los indicadores que ya fueron definidos anteriormente, son los que permitieron definir el Framework que ofrece un mejor rendimiento para el desarrollo de aplicaciones web.

Las herramientas que se utilizaron para el análisis son las siguientes:

Administrador de tareas de Windows y GPU-Z.

### Definición de Parámetros de comparación

El parámetro señalado, junto con cada indicador, fueron seleccionados por los autores de este trabajo de investigación, estos permiten definir el rendimiento de los Frameworks AngularJS y EmberJS, mediante los cuales se realizó el estudio comparativo. Se describe de forma breve en la Tabla 3-2.

Tabla 3-2: Definición de Indicadores.

|  |  |  |
| --- | --- | --- |
| **Parámetro** | **Indicador** | **Descripción** |
| Rendimiento | CPU | Porcentaje de uso de procesador por la aplicación |
| Memoria RAM | Cantidad de memoria que emplea la aplicación para su funcionamiento |
| Disco | Promedio de bytes por segundo a los que tuvo acceso (lectura y escritura) la aplicación |
| Red | Promedio de bytes por segundo Trasferidos (envío y recepción) |
| Tarjeta de video | Cantidad de uso de Tarjeta Gráfica por los Frameworks al dibujar los componentes que se visualizan. |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014

**Uso de CPU**

Al momento de ejecutarse una aplicación, se requiere el uso del CPU, para que esta acción sea posible, cada uno de los prototipos necesitará de este recurso.

**Cantidad de Memoria RAM**

Es necesario que la aplicación sea cargada en memoria para la ejecución de la misma, es por ello que se ha considerado este indicador.

**Uso de Disco**

Aunque sus valores sean muy reducidos su importancia radica en la cantidad de bytes leídos y escritos por el proceso.

**Uso de Red**

Al trabajar con aplicaciones web, uno de los recursos importantes a tomar en cuenta son los bytes por segundo transferidos (envío y recepción) a través de la red por dicha aplicación.

**Uso de Tarjeta Gráfica**

Se requiere del uso de esta tarjeta que permita graficar los componentes que son mostrados al usuario final, ya que es el medio por el cual dicho usuario interactúa con la aplicación.

### Descripción de los módulos de pruebas

El funcionamiento de los módulos definidos para cada una de las pruebas, permiten comprobar el rendimiento de la aplicación desarrollada con los Frameworks AngularJS y EmberJS.Dado que JavaScript se ejecuta en el lado del cliente, se definió separar mediante servicios REST, al Frontend del Backend, y así poder realizar el análisis únicamente del primero.

El Backend de la aplicación está desarrollado en Python, es la encargada de proveer los servicios REST en formato JSON para ser consumidos por el Frontend. En la parte del Frontend, se desarrolló un prototipo del módulo de ventas con cada uno de los Frameworks, que forma parte de la aplicación final, esto para la realización de las respectivas pruebas y su posterior análisis.

***Módulo de ventas***

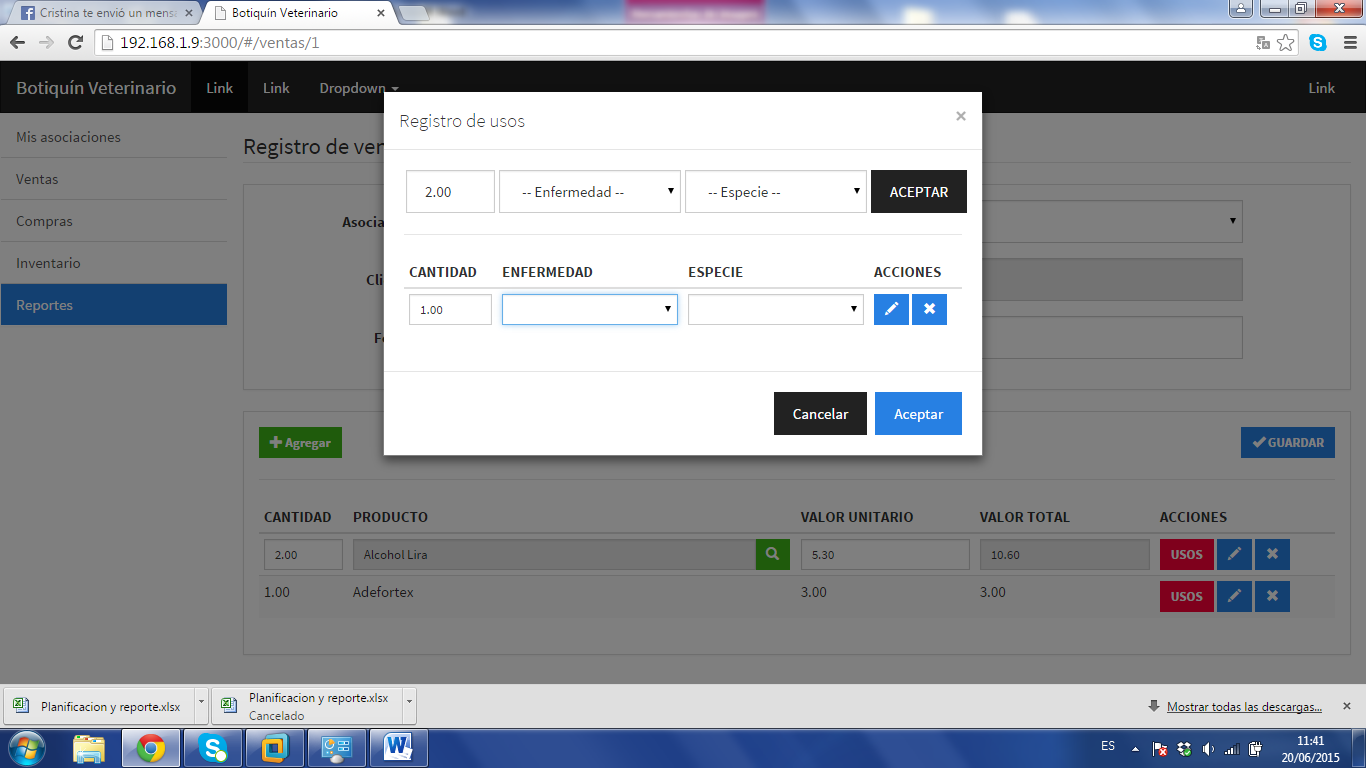
Este módulo permite a los técnicos encargados de las asociaciones, listar y registrar sus ventas teniendo en cuenta que una venta contiene datos del cliente, fecha, detalles. Cada producto que se registre en una venta debe también especificar para que especie de animal y que enfermedad va a tratarse.

### Desarrollo de los módulos de pruebas

#### Framework AngularJS

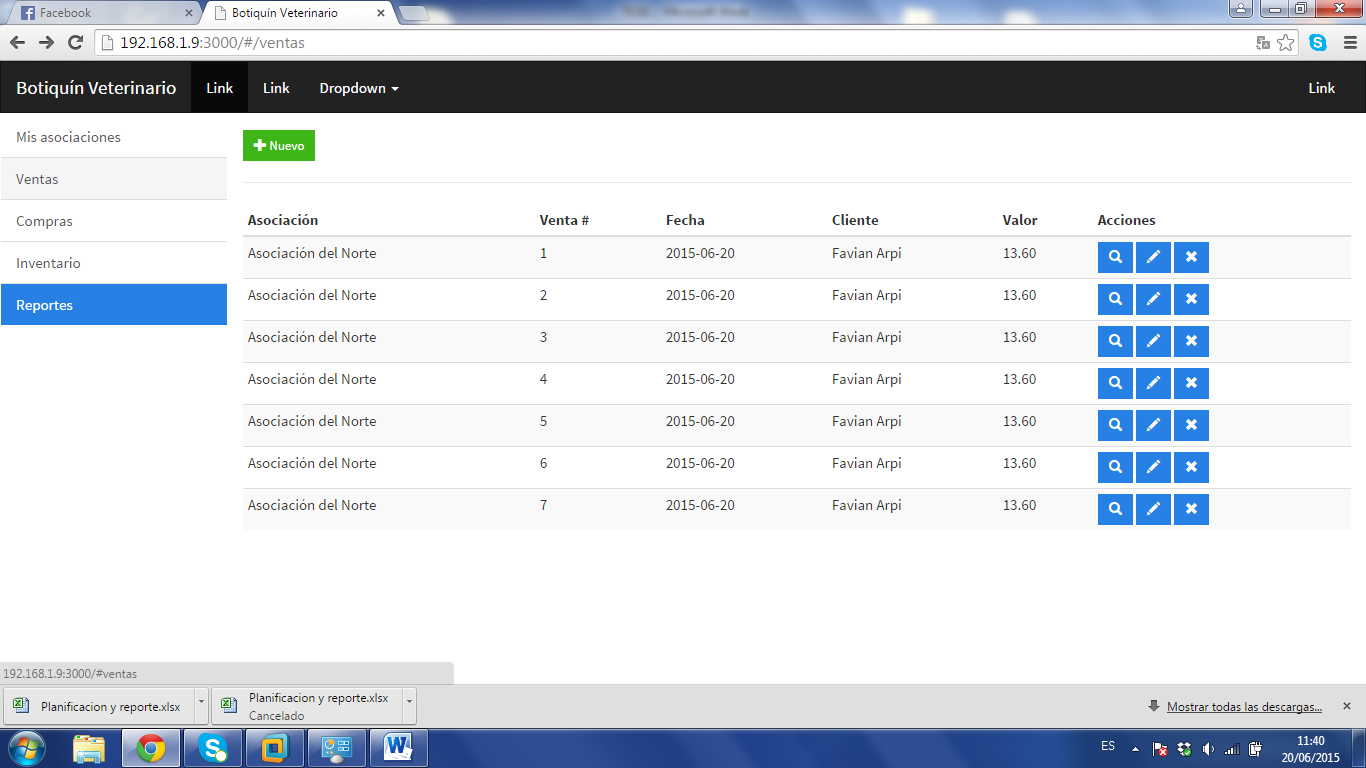
*Módulo de Ventas*

Las figuras 9-2 y 10-2 muestran la interfaz gráfica del prototipo del módulo de ventas desarrollado con el Framework AngularJS.



**Figura 9-2.** Prototipo del Framework AngularJs

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014



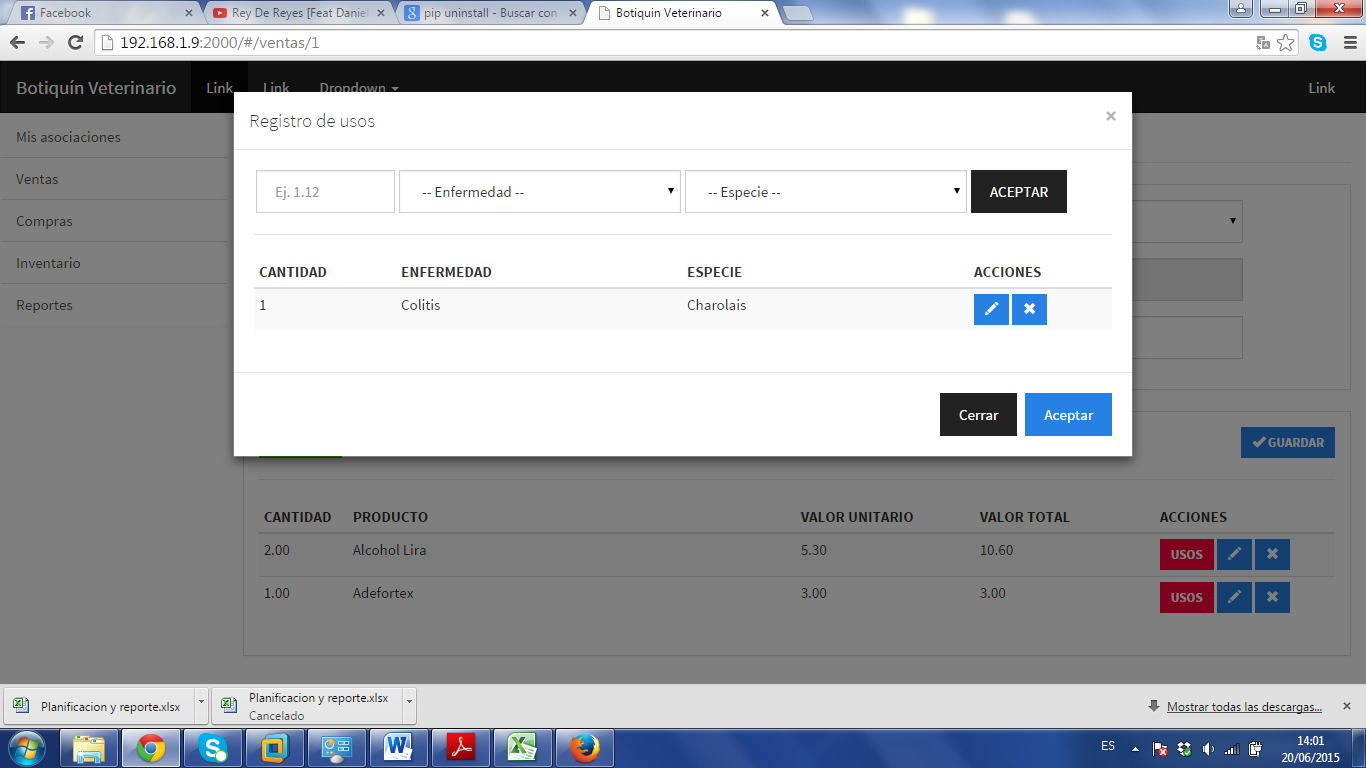
**Figura 10-2.** Datos ingresados en el Prototipo AngularJs

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014

#### Framework EmberJS

*Módulo de Ventas*

Las figuras 11-2 y 12-2 muestran la interfaz gráfica del prototipo del módulo de ventas desarrollado con el Framework EmberJS.



**Figura 11-2.** Prototipo del Framework EmberJS

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014



**Figura 12-2.** Datos ingresados en el Prototipo EmberJS

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014

# CAPITULO III

## MARCO DE RESULTADOS, DISCUSIÓN Y ANÁLISIS DE RESULTADOS

### Análisis Comparativo entre los Framework AngularJS y EmberJS

Los resultados obtenidos por cada uno de los indicadores, permite observar y definir los valores cuantitativos que presentan los prototipos al ser ejecutados.

Las características de hardware utilizado para la realización de las pruebas de los prototipos se describen en la Tabla 4-3.

Tabla 4-3: Características de Hardware

|  |  |
| --- | --- |
| **Características** | **Descripción** |
| Microprocesador | Core i7-2600 3.40GHz |
| RAM | 4 GB |
| Tarjeta Gráfica | 1696 MB |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Indicador 1: CPU

Se realizaron ocho pruebas para cada Framework, cuatro con el método GET que permite obtener datos del API REST, y cuatro con el método POST para enviar datos. Los resultados de las pruebas están expresados en porcentaje de uso del CPU, utilizado por los Frameworks que son el objeto de estudio como se muestra en la Tabla 5-3, para obtener los resultados se utilizó el administrador de tareas de Windows. **ANEXO A**

**Tabla 5-3:**Resultados de CPU en %

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicador** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| **%**CPU | 1,8 | 0,2 | 2,2 | 0,6 |
| 1,4 | 0,3 | 2,3 | 0,5 |
| 1,6 | 0,3 | 1,9 | 0,6 |
| 1,5 | 0,3 | 2,4 | 0,5 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

El factor más importante para calcular el rendimiento del CPU, es la velocidad del reloj, que a mayor velocidad permite ejecutar una mayor cantidad de operaciones por segundo, provocando que el CPU trabaje más y viceversa.

Para poder obtener el valor cuantitativo se realizó lo siguiente.

Definir la cantidad total de GHz a la que trabaja el CPU: 3,40 GHz

Se realizó una regla de tres simple para la conversión a GHz y Hz, como se muestra en las Tablas 6-3 y 7-3.

3,40 = 100%

**Tabla 6-3:**Resultados de CPU expresados en GHz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicadores** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| **CPU**  **(GHz)** | 0,0612 | 0,0068 | 0,0748 | 0,0204 |
| 0,0476 | 0,0102 | 0,0782 | 0,017 |
| 0,0544 | 0,0102 | 0,0646 | 0,0204 |
| 0,051 | 0,0102 | 0,0816 | 0,017 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 7-3:**Resultados de CPU expresados en Hz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicadores** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| **CPU**  **(Hz)** | 61200000 | 6800000 | 74800000 | 20400000 |
| 47600000 | 10200000 | 78200000 | 17000000 |
| 54400000 | 10200000 | 64600000 | 20400000 |
| 51000000 | 10200000 | 81600000 | 17000000 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Indicador 2: Memoria RAM

Se realizaron ocho pruebas para cada Framework, cuatro con el método GET que permite obtener datos del API REST, y cuatro con el método POST para enviar datos.

Los resultados de las pruebas, están expresados en la cantidad de MB de Memoria RAM utilizada por los Frameworks, que son el objeto de estudio, como se presenta en la Tabla 8-3, se obtuvo los resultados con la ayuda del administrador de tareas de Windows. **ANEXO A.**

**Tabla 8-3:**Resultados Memoria RAM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicador** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| **Memoria RAM**  **(MB)** | 77,0 | 86,6 | 91,7 | 120,7 |
| 77,7 | 87,1 | 91,1 | 120,5 |
| 79,1 | 82,7 | 89,1 | 118,0 |
| 80,8 | 84,2 | 90,8 | 118,6 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Indicador 3: Disco

Se realizó ocho pruebas para cada Framework, cuatro con el método GET que permite obtener datos del API REST, y cuatro con el método POST para enviar datos.

Los resultados de las pruebas, están expresados en promedio de kB/s a los que tuvo acceso el proceso (lectura y escritura) de Disco, utilizada por los Frameworks como se muestra en la Tabla 9-3, se obtuvo los resultados con la ayuda del administrador de tareas de Windows. **ANEXO A.**

**Tabla 9-3:**Resultados Disco

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicador** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| kB/s Disco | 972 | 320 | 960 | 264 |
| 976 | 272 | 960 | 316 |
| 728 | 268 | 690 | 312 |
| 508 | 268 | 512 | 251.25 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Indicador 4: Red

Se realizó ocho pruebas para cada Framework, cuatro con el método GET que permite obtener datos del API REST, y cuatro con el método POST para enviar datos. Los resultados de las pruebas, están expresados en promedio de b/s transferidos por el proceso (envío y recepción) en la Red, utilizada por los Frameworks como se muestra en la Tabla 10-3, se obtuvo los resultados con la ayuda del administrador de tareas de Windows. **ANEXO A.**

**Tabla 10-3:**Resultados Red

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicador** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| b/s Red | 12410 | 8880 | 51049 | 4111 |
| 12362 | 8629 | 51117 | 4216 |
| 12379 | 5750 | 51033 | 8432 |
| 12393 | 5751 | 50562 | 5619 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Indicador 5: Tarjeta Gráfica

Este indicador puede ser medido al dibujar y visualizar los componentes gráficos que se obtienen como resultado de una petición por parte del usuario, motivo por el cual no permite obtener resultados con la ejecución del método POST, quien únicamente envía datos.

Los siguientes resultados están expresados en cantidad de MB de Tarjeta Gráfica utilizada por los Frameworks, estos se han obtenido de la ejecución del método GET de las distintas pantallas del prototipo del módulo de ventas al azar: Lista de ventas, cliente, producto y usos. Para realizar las pruebas del indicador se utilizó la herramienta GPU-Z.

GPU-Z al ser una herramienta que calcula el total de Tarjeta Gráfica utilizada por el computador,se realizó lo siguiente:

Se midió el total de Tarjeta Gráfica utilizada sin ejecutar los prototipos, de igual manera cuando éstos fueron ejecutados. Con los datos obtenidos en la Tabla 11-3, **ANEXO A**, se calculó la diferencia entre ellos, obteniendo la cantidad real utilizada por cada Framework como muestra la Tabla 12-3.

**Tabla 11-3:**Resultados Tarjeta Gráfica

|  |  |  |
| --- | --- | --- |
| **Indicador** | **AngularJS** | **EmberJS** |
| **GET** | **GET** |
| MB Tarjeta Gráfica | 64 | 65 |
| 59 | 63 |
| 62 | 64 |
| 60 | 62 |

**Fuente:** PEÑA, J., CAMBISACA, M., 2015

Total de Mega Bytes de tarjeta Gráfica utilizada por el computador sin ejecutar los prototipos. **ANEXO B**

51 MB

**Tabla 12-3:**Resultados Tarjeta Gráfica Utilizada por cada Framework.

|  |  |  |
| --- | --- | --- |
| **Indicador** | **AngularJS** | **EmberJS** |
| **GET** | **GET** |
| MB Tarjeta Gráfica | 13 | 14 |
| 8 | 12 |
| 11 | 13 |
| 9 | 11 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

### Análisis de Resultados

Se aplicó la fórmula de la media () para la obtención de resultados, por indicador y método de cada Framework, como se muestra en la Tabla 13-3.

**Tabla 13-3:** Resultados por indicador y método de cada Framework

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicadores** | **AngularJS** | | **EmberJS** | |
| **GET** | **POST** | **GET** | **POST** |
| **CPU** | 61200000 | 6800000 | 74800000 | 20400000 |
| 47600000 | 10200000 | 78200000 | 17000000 |
| 54400000 | 10200000 | 64600000 | 20400000 |
| 51000000 | 10200000 | 81600000 | 17000000 |
| **GHz** | **53550000** | **9350000** | **74800000** | **18700000** |
| **Memoria RAM** | 77,0 | 86,6 | 91,7 | 120,7 |
| 77,7 | 87,1 | 91,1 | 120,5 |
| 79,1 | 82,7 | 89,1 | 118,0 |
| 80,8 | 84,2 | 90,8 | 118,6 |
| **MB** | **78,65** | **85,15** | **90,675** | **119,45** |
| **Disco** | 972 | 320 | 960 | 264 |
| 976 | 272 | 960 | 316 |
| 728 | 268 | 690 | 312 |
| 508 | 268 | 512 | 251.25 |
| **kB/s** | **796** | **282** | **780,5** | **285,8125** |
| **Red** | 12410 | 8880 | 51049 | 4111 |
| 12362 | 8629 | 51117 | 4216 |
| 12379 | 5750 | 51033 | 8432 |
| 12393 | 5751 | 50562 | 5619 |
| **b/s** | **12386** | **7252,5** | **50940,25** | **5594,5** |
| **Tarjeta Gráfica** | 13 |  | 14 |  |
| 8 |  | 12 |  |
| 11 |  | 13 |  |
| 9 |  | 11 |  |
| **MB** | 10,25 |  | 12,5 |  |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

De igual manera se realizó la media entre los métodos GET y POST de cada indicador, para obtener un resultado final de las pruebas por cada indicador y Framework.

Para la obtención de resultados finales en porcentajes se realizó una regla de tres simple, se tomó como 100% de cada indicador, al valor máximo obtenido de las pruebas realizadas, como se detalla a continuación, y se muestran los resultados en la Tabla 14-3 y la Figura 13-3.

**CPU**

78200000 Hz

78200000 =100%

**RAM**

121 MB

121 = 100%

**Disco**

976kB/s

976 =100%

**Red**

51117b/s

51117 =100%

**Tarjeta Gráfica**

14 MB

14=100%

**Tabla 14-3:** Resultados y Porcentajes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Indicadores** | **Valor Máximo** | **AngularJS** | | **EmberJS** | |
| **Promedio GET-POST** | **%** | **Promedio GET-POST** | **%** |
| CPU | 78200000 | 31450000 | 40,22 | 46750000 | 59,78 |
| Memoria RAM | 121 | 81,9 | 67,66 | 105,06 | 86,83 |
| Disco | 976 | 539 | 55,23 | 533,16 | 54,63 |
| Red | 51117 | 9819,25 | 19,21 | 28267,38 | 55,3 |
| Tarjeta Gráfica | 14 | 10,25 | 73,21 | 12,5 | 89,29 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Figura 13-3.** Resultados, Consumo de Recursos.

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

### Interpretación de los resultados

Mediante el establecimiento de los resultados podemos definir que el Framework AngularJS presenta un uso de CPU del 40,22% siendo menor al Framework EmberJS, que responde con un consumo de CPU del 59,78%.

Para el uso de memoria RAM por parte del Framework AngularJS, se obtuvo el 67,66% que es menor al de EmberJS, respondiendo con un 86,83%, por lo que se puede definir que AngularJS requiere de una menor cantidad de Memoria RAM para realizar las mismas acciones que EmberJS.

AngularJS presenta un 55,23% en el proceso de acceso (lectura y escritura) de Disco, mientras que EmberJS obtuvo un 54,63%, lo que demuestra que éste segundo, lee y escribe menor cantidad de kB/s en el disco al ejecutar sus procesos.

Se observa que el Framework AngularJS responde con el 19,21% en el indicador Red, siendo menor al de EmberJS, que es del 55,3%, demostrando que AngularJS transfiere en la Red una menor cantidad de B/s cumpliendo las mismas funciones que EmberJS.

El uso de la Tarjeta Gráfica para dibujar y visualizar los componentes del Framework AngularJS obtuvo un 73,21%, siendo menor al resultado obtenido por EmberJS que es del 89,29%, lo que demuestra que AngularJS utiliza menor cantidad de tarjeta gráfica para la visualización de sus componentes.

### Comprobación de la Hipótesis

***Hipótesis:***

**H1:** El desarrollo de aplicaciones web con el Framework JavaScript MVC AngularJS, permitirá obtener mayor rendimiento que con el Framework EmberJS.

**Tabla 15-3:** Porcentajes de Consumo o Utilización de Recursos.

|  |  |  |
| --- | --- | --- |
| **Indicadores** | **AngularJS** | **EmberJS** |
| **%** | **%** |
| CPU | 40,22 | 59,78 |
| Memoria RAM | 67,66 | 86,83 |
| Disco | 55,23 | 54,63 |
| Red | 19,21 | 55,3 |
| Tarjeta Gráfica | 73,21 | 89,29 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2014

Para obtener el porcentaje total de cada Framework, se asigna una ponderación del 20% a todos los indicadores, considerando que cada indicador tiene un igual grado de importancia, esto permite obtener como resultado el 100% que es el puntaje más alto de consumo o utilización de recursos, como muestra la La Tabla 16-3 en base a la Tabla 15-3. El porcentaje final de consumo o utilización de recursos se presenta en la Tabla 17-3.

**Tabla 16-3:** Porcentajes de Consumo de Recursos, por ponderación de indicador.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Indicadores** | **Ponderación** | **AngularJS** | | **EmberJS** | |
| **%** | **%Consumo** | **%** | **% Consumo** |
| CPU | 20% | 40,22 | 8,044 | 59,78 | 11,956 |
| Memoria RAM | 20% | 67,66 | 13,532 | 86,83 | 17,366 |
| Disco | 20% | 55,23 | 11,046 | 54,63 | 10,926 |
| Red | 20% | 19,21 | 3,842 | 55,3 | 11,06 |
| Tarjeta Gráfica | 20% | 73,21 | 14,642 | 89,29 | 17,858 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 17-3:** Porcentaje Final, Consumo o Utilización de Recursos.

|  |  |
| --- | --- |
| **Consumo** | |
| **Framework** | **Porcentaje** |
| **AngularJS** | 51,11 |
| **EmberJS** | 69,17 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

Debido a que, a menor consumo o utilización de recursos, se obtiene un mayor rendimiento y viceversa, se define que el consumo y rendimiento son inversamente proporcionales entre sí. Por lo tanto el resultado del rendimiento para cada Framework se muestra en la Tabla 18-3, y en la figura 14-3.

**Tabla 18-3:** Porcentajes de Rendimiento

|  |  |
| --- | --- |
| **Rendimiento** | |
| **Framework** | **Porcentaje** |
| **AngularJS** | 48,89 |
| **EmberJS** | 30,83 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Figura 14-3.** Resultados de Rendimiento.

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En referencia a las Tablas 16-3, 17-3, 18-3 y mediante observación directa se concluye que el Framework AngularJS, tiene un porcentaje de rendimiento del 48,89%, superando en un 18,06% al Framework EmberJS, que posee un rendimiento del 30,83%. Por lo tanto se concluye que la hipótesis H1 es verdadera.

# CAPITULO IV

## PROPUESTA DE DESARROLLO DEL SISTEMA DE CONTROL DE

## BOTIQUINES (SICOBO)

### Visión y Alcance

El principal objetivo es desarrollar un sistema web diseñado para automatizar los procesos de recolección de datos en el programa “Botiquines Veterinarios”, que lleva a cabo el l Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP). Para llegar de la mejor manera a los objetivos planteados en el desarrollo de un sistema informático es necesario seguir la guía de una metodología de desarrollo de software.

El sistema SICOBO está realizado tomando en cuentas las pautas que dicta la metodología MSF (Microsoft Solutions Framework), por lo cual en este capítulo se describen cada una de sus fases en las que se explica desde la visión y alcance del proyecto, hasta llegar a la realización de estabilización y sus respectivas pruebas, para su posterior implementación por parte del personal del MAGAP.

#### Definición del Problema.

El MAGAP, es una institución encargada de controlar, normar, facilitar, y evaluar la gestión de la producción agrícola, ganadera, acuícola y pesquera del país; promoviendo acciones que permitan el desarrollo rural y propicien el crecimiento sostenible de la producción y productividad del sector impulsando al desarrollo de productores.

En la provincia de Morona Santiago esta institución ha impulsado un proyecto denominado “Botiquines Veterinarios“, en el cual asociaciones que son conformadas dentro de los cantones de la provincia reciben un paquete de insumos veterinarios tales como medicinas, vitaminas, instrumental veterinario, para que dichas asociaciones emprendan un negocio propio y presten servicios a agricultores y ganaderos de su sector.

Sin embargo, este programa requiere que cada cierto tiempo se elaboren reportes estadísticos para determinar la eficiencia del proyecto, esto resulta complejo y laborioso de realizarlo debido a que no existe un modelo a seguir por las asociaciones que les permita registrar los datos de una manera correcta.

* + 1. Perfiles de Usuario.

*Administrador*

Persona con título de Ingeniero en sistemas informáticos y conocimientos en Python, HTML, CSS, JavaScript y PostgreSql.

Funciones:

* Acceso libre al sitio administrativo del sistema.
* Crear registros nuevos.
* Modificar registros.
* Eliminar registros.

*Director.*

Persona con cargo institucional a quien se reporta datos desde cada una de las asociaciones de la provincia.

Funciones:

* Acceso a registros de interés.
* Generación de reportes estadísticos.
* Visualización de Reportes.
* Impresión de Reportes.

*Técnico:*

Persona con cargo institucional encargada de controlar y recolectar datos de las asociaciones a su cargo.

Funciones:

* Creación de registros de ventas, compras e inventario de asociaciones.
* Edición de registros.
* Eliminación de registros
  + 1. Ámbito del Proyecto.

El sistema se desarrolla para El Ministerio de Agricultura, Ganadería, Acuacultura y Pesca en la provincia de Morona Santiago y lleva por nombre SICOBO (Sistema de Control de Botiquines), mismo que en un sistema para automatizar y facilitar el control y recolección de datos del programa Botiquines Veterinarios.

2. 1. 5. Herramientas a Utilizar.

Las herramientas usadas en el desarrollo de este proyecto están definidas en la Tabla 19-4.

**Tabla 19-4:** Herramientas a Utilizar

| **HERRAMIENTA** | **CARACTERÍSTICA** |
| --- | --- |
| Python | Lenguaje de programación |
| Django | Framework para crear aplicaciones web en python |
| Continúa | |
| Django Rest Framework | Framework para el desarrollo de api-rest en python |
| Sublime Text | Editor de código |
| Angular JS | Framework MVC en el lado del cliente. |
| Ember JS | Framework MVC en el lado del cliente. |
| JQuery | Librería de JavaScript. |
| Bootstrap | Framework CSS. |
| Microsoft Project. | CASE para Calendarización y Cálculo de Presupuesto. |
| Vertabelo | Modelado de bases de datos. |
| PostgreSQL | Base de datos. |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

* + 1. Objetivos del Proyecto.

*Objetivo General*

* Desarrollar un sistema de control de Botiquín Veterinario para el Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP) – Morona Santiago.

*Objetivos Específicos*

* Diseñar la estructura de base de datos relacional.
* Desarrollar un API-REST para el envío y recepción de datos.
* Desarrollar una aplicación con la tecnología AngularJS.
* Diseñar interfaces de usuario amigables utilizando el Framework Bootstrap
  + 1. Análisis y gestión del riesgo.

Previo al desarrollo de una aplicación es importante realizar un análisis y gestión de riesgos, que lleva el propósito de definir las posibles partes vulnerables que requieren ser tomadas en cuenta para su protección y de esta manera minimizar riesgos y determinar las amenazas que lo provocan.

El listado de riesgos que pueden afectar o retrasar el desarrollo del proyecto, está detallado en la Tabla 20-4.

**Tabla 20-4:** Riesgos

| **Nº** | **Riesgo** | **Categoría** | **Consecuencia** |
| --- | --- | --- | --- |
| 1 | Las computadoras en las que se realiza el desarrollo del proyecto se dañaron y no permiten la recuperación de la información. | Riesgo del proyecto | Cancelación del proyecto |
| 2 | El Ingeniero de asesoramiento del proyecto ya no pueda continuar | Riesgo del proyecto | Retraso en el proyecto |
| 3 | Miembro del equipo de desarrollo deja o renuncia al proyecto. | Riesgo del negocio | Retraso en el proyecto |
| 4 | Se vuelva complejo el aprendizaje de la herramienta de desarrollo. | Riesgo técnico | Retraso en el proyecto |
| 5 | Requerimientos de software mal interpretados | Riesgo técnico | Planificación inadecuada |
| 6 | Se recorta el plazo para la entrega del proyecto. | Riesgo técnico | Software inadecuado |
| 7 | Cambio continuo de los requerimientos de software | Riesgo del proyecto | Demoras en la programación |
| Continúa | | | |
| 8 | El sistema no tiene un buen diseño de experiencia de usuario | Riesgo de negocio | Pérdida de recursos, rediseño de interfaces. |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

Para determinar el impacto que tendrá cada uno de los riesgos mencionados anteriormente, debe seguirse ciertos parámetros que se indican en la Tabla 21-4.

**Tabla 21-4:** Determinación de Impacto

| **Impacto** | **Impacto Técnico** | **Retraso** | **Impacto del Costo** | **Valor** |
| --- | --- | --- | --- | --- |
| Bajo | Ligero efecto al desarrollo del proyecto | 5 días | <1% | 1 |
| Moderado | Moderado efecto al desarrollo del proyecto | 10 días | <5% | 2 |
| Alto | Severo efecto al desarrollo del proyecto | 15 días | <10% | 3 |
| Crítico | Proyecto no puede ser culminado | 20 días | >=10% | 4 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

Para determinar la probabilidad de ocurrencia de cada uno de los riesgos mencionados anteriormente se debe seguir ciertos parámetros que se indican en la Tabla 22-4.

**Tabla 22-4:** Probabilidad de que el riesgo ocurra.

|  |  |  |
| --- | --- | --- |
| **Rango de Probabilidad** | **Descripción** | **Valor** |
| 1% y <= 33% | Baja | 1 |
| >33% y <= 66% | Media | 2 |
| >67% y <= 99% | Alta | 3 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

Las tablas 23-4 y 24-4, indican como determinar y categorizar la exposición de cada uno de los riesgos.

**Tabla 23-4:** Determinación de exposición al riesgo.

|  |  |  |
| --- | --- | --- |
| **Color** | **Exposición** | **Valor** |
| Verde | Baja | 1 o 2 |
| Amarillo | Media | 3 o 4 |
| Rojo | Alta | Mayor a 6 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 24-4:** Código de colores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Impacto |  |  |
| Probabilidad | Bajo=1 | Moderado=2 | Alto=3 | Crítico=4 |
| Baja | 3 | 6 | 9 | 12 |
| Media | 2 | 4 | 6 | 8 |
| Alta | 1 | 2 | 3 | 4 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *Desarrollo del análisis de Riesgos.*

El investigador define una probabilidad de ocurrencia e impacto en cada riesgo, tomando en cuenta los parámetros indicados en las tablas 21-4 y 22-4, luego debe calcularse la exposición y prioridad tomando como referencia cada uno de los parámetros de las tablas 23-4 y 24-4, como esta mostrado en la tabla 25-4.

**Tabla 25-4:** Análisis de Riesgo

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Id**  **Riesgo** | **PROBABILIDAD** | | | **IMPACTO** | | **EXPOSICIÓN** | | **Prioridad** |
| **%** | **Valor** | **Probabilidad** | **Impacto** | **Valor** | **Valor** | **Exposición** |
| R1 | 10% | 1 | Baja | Critico | 4 | 4 | Media | 3 |
| R2 | 50% | 2 | Media | Moderado | 2 | 4 | Media | 3 |
| R3 | 50% | 2 | Media | Critico | 4 | 8 | Alta | 2 |
| R4 | 5% | 1 | Baja | Moderado | 2 | 2 | Baja | 4 |
| R5 | 75% | 3 | Alta | Alto | 3 | 9 | Alta | 1 |
| R6 | 70% | 3 | Alta | Alto | 3 | 9 | Alta | 1 |
| R7 | 90% | 3 | Alta | Alto | 3 | 9 | Alta | 1 |
| R8 | 50% | 2 | Media | Moderado | 2 | 4 | Media | 3 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *Línea de Corte.*

La tabla 26-4 ilustra los riesgos que requieren la más urgente gestión debido a que son los que tienen alta exposición, la línea de corte divide a estos del resto.

**Tabla 26-4:** Línea de Corte

|  |  |
| --- | --- |
| **Identificación** | **Exposición** |
| R5 | 9 |
| R6 | 9 |
| R7 | 9 |
| R3 | 8 |
| R1 | 4 |
| R2 | 4 |
| R8 | 4 |
| R4 | 2 |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

Las tablas 27-4 hasta la tabla 30-4, permiten observar en detalle los riegos considerados prioritarios, dando a conocer las causas y consecuencias del mismo, y sobre todo como gestionar el riesgo encontrado.

**Tabla 27-4:** Riesgo 5

|  |  |  |  |
| --- | --- | --- | --- |
| **HOJA DE GESTION DEL RIESGO** | | | |
| **ID DEL RIESGO:** R5 | | **FECHA:** | |
| **Probabilidad:** Alta  **Valor:** 3 | **Impacto:** Alto  **Valor:** 3 | **Exposición:** Alta  **Valor:** 9 | **Prioridad:** 1 |
| **DESCRIPCION:** Requerimientos de software mal interpretados | | | |
| **REFINAMIENTO:**  **Causas:** Debido a cierta inexperiencia por parte de los desarrolladores no se logró comprender a plenitud las necesidades del cliente.  **Consecuencias:** Retraso en el proyecto, inconformidad por parte del cliente con el producto final. | | | |
| **REDUCCIÓN:** Preparar al equipo de desarrollo sobre técnicas para definir requerimientos de software, además de mantener constante comunicación con el cliente. | | | |
| **SUPERVISIÓN:** Planificar pequeñas entregas al cliente para constatar que el proyecto sigue el camino que el cliente desee. | | | |
| **GESTIÓN:** Mantener con el cliente constante comunicación y hacer entregas pequeñas del proyecto. | | | |
| **ESTADO ACTUAL:**  Fase de reducción iniciada  Fase de Supervisión iniciada  Gestionando el Riesgo | | | |
| **RESPONSABLES:** Jorge Peña, Milton Cambisaca | | | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 28-4:** Riesgo 6

|  |  |  |  |
| --- | --- | --- | --- |
| **HOJA DE GESTION DEL RIESGO** | | | |
| **ID DEL RIESGO:** R6 | | **FECHA:** | |
| **Probabilidad:** Alta  **Valor:** 3 | **Impacto:** Alto  **Valor:** 3 | **Exposición:** Alta  **Valor:** 9 | **Prioridad:** 1 |
| **DESCRIPCION:** Se recorta el plazo para la entrega del proyecto | | | |
| **REFINAMIENTO:**  **Causas:** Por motivos ajenos al proyecto se requiere tener un producto terminado en un tiempo menor al establecido.  **Consecuencias:**   * Trabajo bajo presión en el equipo de desarrollo. * No se cumple con todos los requerimientos. | | | |
| **REDUCCIÓN:** Mantener una planificación y seguirla, tratando de adelantar las tareas, además de aumentar las horas de trabajo. | | | |
| **SUPERVISIÓN:** Usar software que permita asignar y controlar el cumplimiento de tareas. | | | |
| **GESTIÓN:** Usar herramientas para facilitar el desarrollo en equipo, y dar prioridad a requerimientos más importantes. | | | |
| **ESTADO ACTUAL:**  Fase de reducción iniciada  Fase de Supervisión iniciada  Gestionando el Riesgo | | | |
| **RESPONSABLES:** Jorge Peña, Milton Cambisaca | | | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 29-4:** Riesgo 7

|  |  |  |  |
| --- | --- | --- | --- |
| **HOJA DE GESTION DEL RIESGO** | | | |
| **ID DEL RIESGO:** R7 | | **FECHA:** | |
| **Probabilidad:** Alta  **Valor:** 3 | **Impacto:** Alto  **Valor:** 3 | **Exposición:** Alta  **Valor:** 9 | **Prioridad:** 1 |
| **DESCRIPCION:** Cambio continuo de los requerimientos de software | | | |
| **REFINAMIENTO:**  **Causas:** Durante el desarrollo del proyecto el cliente quiere añadir nuevas funcionalidades al sistema.  **Consecuencias:**   * Retraso en los tiempos definidos para el proyecto. * El costo final del proyecto aumenta. * Puede llegar a incumplirse algunas funcionalidades o requerimientos del proyecto. | | | |
| **REDUCCIÓN:** Seguir técnicas de ingeniería de requerimientos para la correcta definición de los mismos. | | | |
| **SUPERVISIÓN:** Tomar al cliente como un miembro más del equipo de trabajo para resolver dudas y establecer claramente los requerimientos. | | | |
| **GESTIÓN:** Definir con el cliente que los nuevos requerimientos incurren en mayor costo y mayor tiempo. | | | |
| **ESTADO ACTUAL:**  Fase de reducción iniciada  Fase de Supervisión iniciada  Gestionando el Riesgo | | | |
| **RESPONSABLES:** Jorge Peña, Milton Cambisaca | | | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 30-4:** Riesgo 3

|  |  |  |  |
| --- | --- | --- | --- |
| **HOJA DE GESTION DEL RIESGO** | | | |
| **ID DEL RIESGO:** R3 | | **FECHA:** | |
| **Probabilidad:** Media  **Valor:** 2 | **Impacto:** Critico  **Valor:** 4 | **Exposición:** Alta  **Valor:** 8 | **Prioridad:** 2 |
| **DESCRIPCION:** Miembro del equipo de desarrollo deja o renuncia al proyecto. | | | |
| **REFINAMIENTO:**  **Causas:** Un miembro del equipo muestra desinterés o tiene otras ocupaciones más importantes que le impiden seguir en el proyecto.  **Consecuencias:** Mal ambiente de trabajo, retraso o no puede llegar a culminarse el proyecto. | | | |
| **REDUCCIÓN:** Generar un buen ambiente de trabajo desde el inicio del proyecto, cumpliendo pequeñas metas para incentivar el desarrollo personal y grupal. | | | |
| **SUPERVISIÓN:** Mantener una constante comunicación con el tutor del trabajo de investigación, quien será el encargado de guiar a los miembros del equipo de trabajo. | | | |
| **GESTIÓN:** Reevaluar el alcance del proyecto además de costos y tiempos de entrega. | | | |
| **ESTADO ACTUAL:**  Fase de reducción iniciada  Fase de Supervisión iniciada  Gestionando el Riesgo | | | |
| **RESPONSABLES:**  Jorge Peña, Milton Cambisaca | | | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Planificación Inicial.

*Definición del Plan Borrador*

Sistema de Control De Botiquín Veterinario “SICOBO”

*Sector Solicitante:* Ministerio de Agricultura, Ganadería, Acuacultura y Pesca - Morona Santiago.

*Equipo de Trabajo*

Es importante definir cada uno de los roles que cumplen los miembros del equipo de trabajo como indica la Tabla 31-4.

**Tabla 31-4:** Equipo de desarrollo y Gestión

|  |  |
| --- | --- |
| **INTEGRANTES** | **CARGOS** |
| Jorge Peña | Gestor de Proyecto, Desarrollador |
| Milton Cambisaca | Gestor de Proyecto, Desarrollador |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Aspectos Generales de la Empresa*

*Nombre de la Empresa:*

MINISTERIO DE AGRICULTURA, GANADERÍA, ACUACULTURA Y PESCA **(MAGAP)**

*Dirección:*

Juan de la Cruz y Guamote, Macas – Ecuador

*Teléfono:*

072701919

*Misión*

"El Ministerio de Agricultura, Ganadería, Acuacultura y Pesca es la institución rectora del multisector, para regular, normar, facilitar, controlar, y evaluar la gestión de la producción agrícola, ganadera, acuícola y pesquera del país; promoviendo acciones que permitan el desarrollo rural y propicien el crecimiento sostenible de la producción y productividad del sector impulsando al desarrollo de productores, en particular representados por la agricultura familiar campesina, manteniendo el incentivo a las actividades productivas en general.” (agricultura.gob.ec)

*Coordinación Zonal*

* *Misión:* Liderar la estrategia territorial del Agro, mediante la planificación, coordinación y control de las actividades que impulsen las necesidades zonales del sector, a fin de constituir un soporte y brindar todas las facilidades a sus usuarios, garantizando la soberanía alimentaria y desarrollo rural. (agricultura.gob.ec)

*Dirección Provincial*

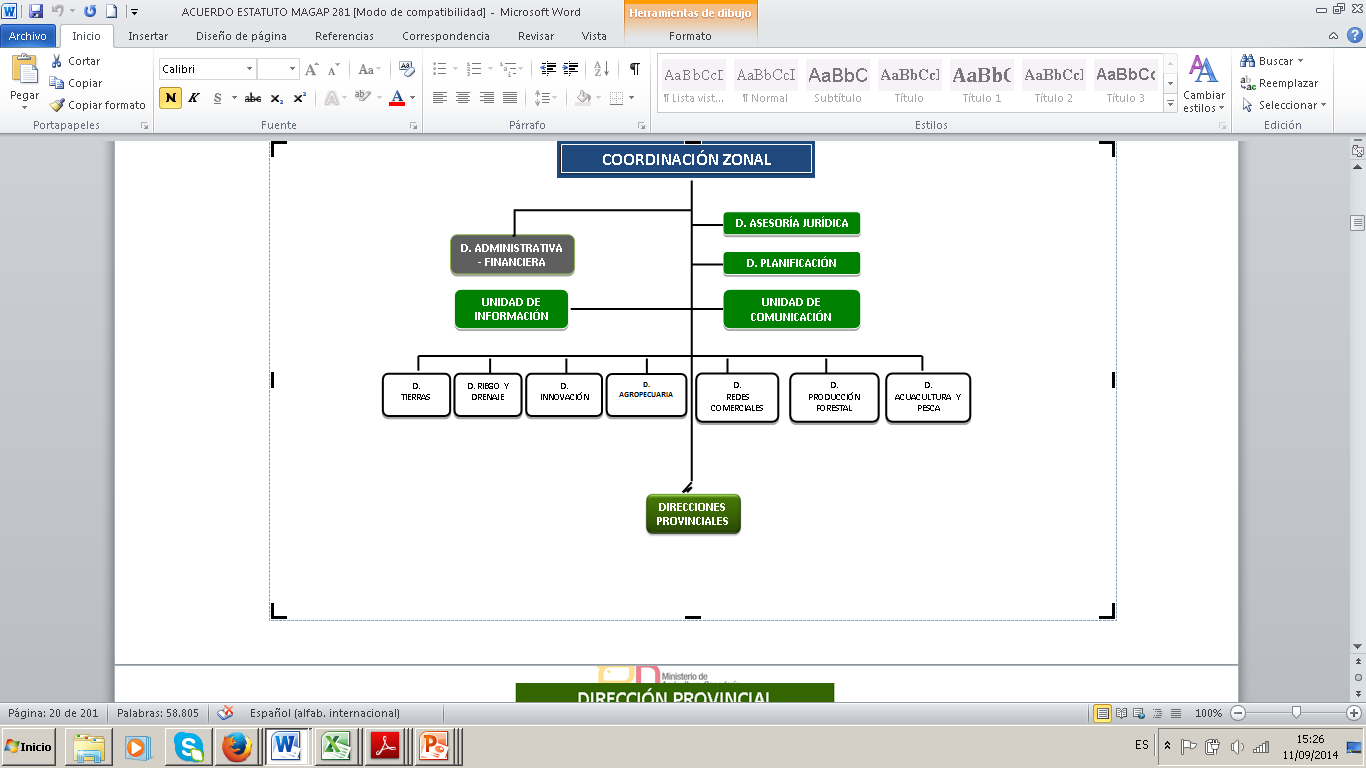
* *Misión:* Gestionar y ejecutar la adecuada distribución de los recursos económicos financieros, la asistencia técnica necesaria y el acompañamiento directo a los usuarios del sector; a fin de proveerlos de las capacidades necesarias para desarrollar las potencialidades productivas del territorio. (agricultura.gob.ec)

*Visión*

"Para el 2020 el Ministerio de Agricultura, Ganadería, Acuacultura y Pesca contará con un modelo de gestión de calidad basado en sistemas de información y comunicación. Que posibiliten la producción de bienes y servicios que garanticen la seguridad alimentaria del país, el crecimiento y desarrollo equitativo, generando valor agregado con rentabilidad económica, equidad social, sostenibilidad ambiental e identidad cultural." (agricultura.gob.ec)

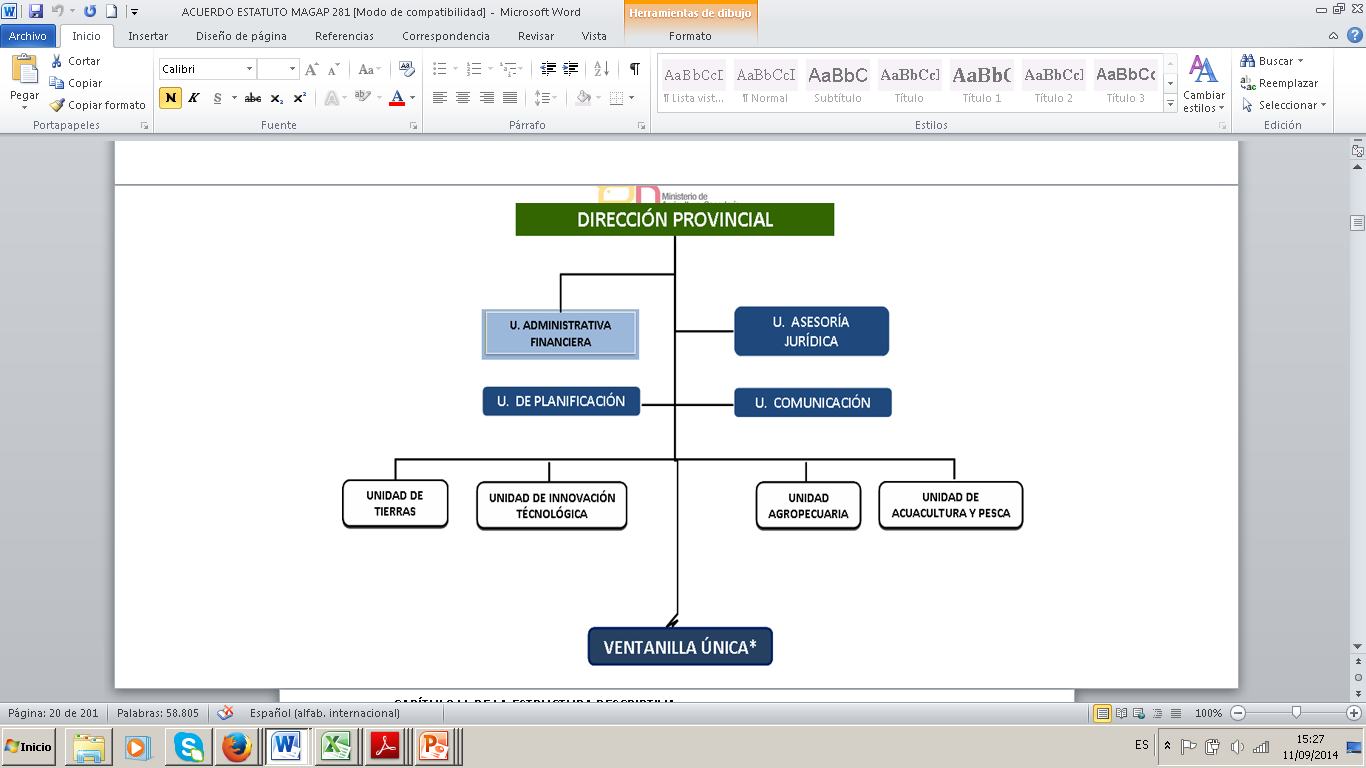
*Orgánico Estructural*

La figura 15-4, muestra la estructura del MAGAP por zona, mientras que la figura 16-4, presenta el orgánico interno – Morona Santiago



**Figura 15-4.** Orgánico Estructural Zonal.

**Fuente:** agricultura.gob.ec



**Figura 16-4.** Orgánico Dirección Provincial.

**Fuente:** http://www.agricultura.gob.ec/

*Generalidades*

*Fuente de Recopilación de la Información*

* El personal encargado del programa “Botiquín Veterinario”, Sr. Miguel Guzmán, será quien provea de cada detalle y requerimientos para el desarrollo del sistema informático.
* Libros, artículos e internet servirán para aprender sobre el uso de las herramientas de desarrollo.

*Técnicas a utilizar*

*Técnicas para la Recopilación de Información.*

*Entrevistas*

* Debe realizarse entrevistas con las personas involucradas en el tema, que se convierten en la fuente de información, para contribuir a cada uno de los requerimientos del proyecto.

*Lluvia de Ideas*

* Es necesario realizar reuniones con el personal involucrado que conoce acerca del tema botiquín veterinario dentro del MAGAP, donde las necesidades y requerimientos saldrán a flote mediante el análisis y discusión que aporte al proyecto, puesto que ellos serán los que van a interactuar con el sistema.

*Técnicas para la representación de actividades del Sistema*

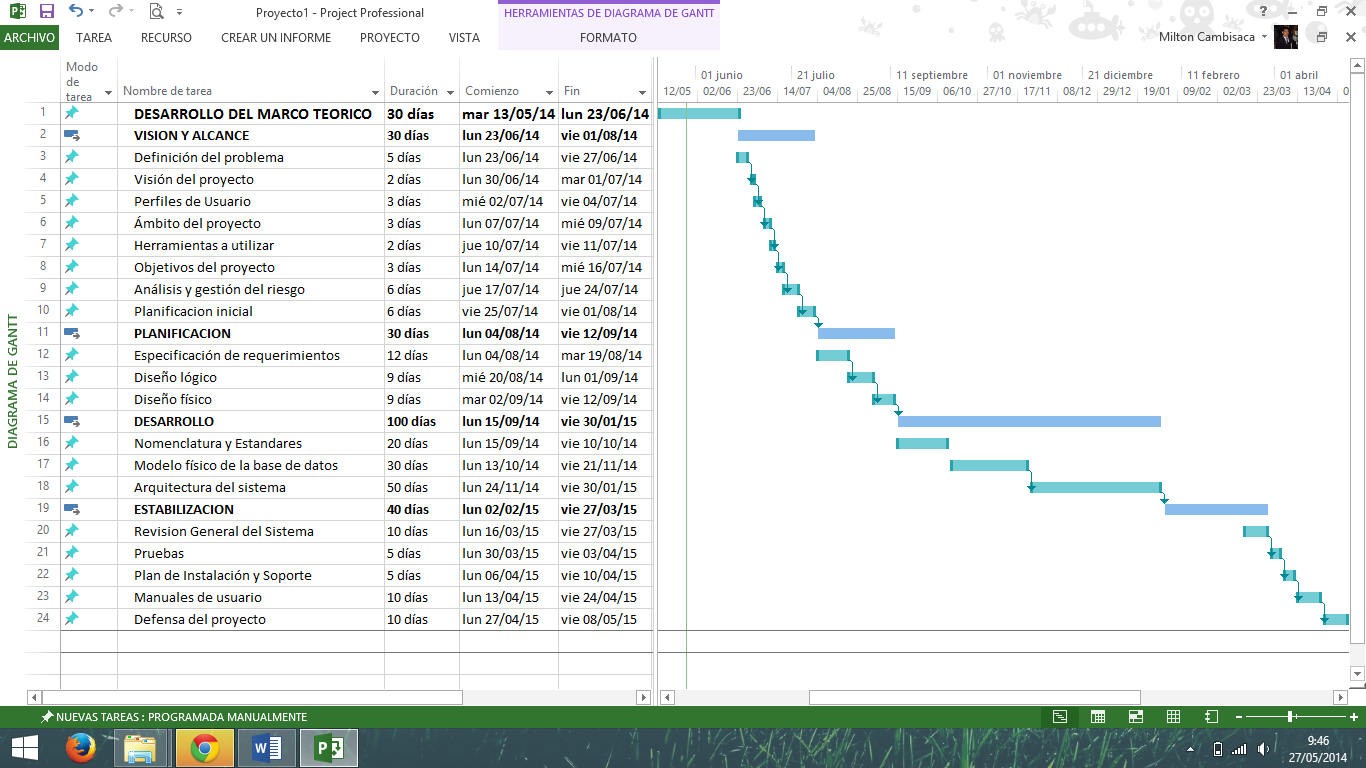
*Técnica de Estructura PIECES*

Permite detectar las necesidades o posibles problemas que puedan presentarse en el desarrollo del sistema, determinando de la manera más viable posible una solución, para cumplir con cada una de las expectativas o exigencias del usuario final.

*Diagrama de Actividades*

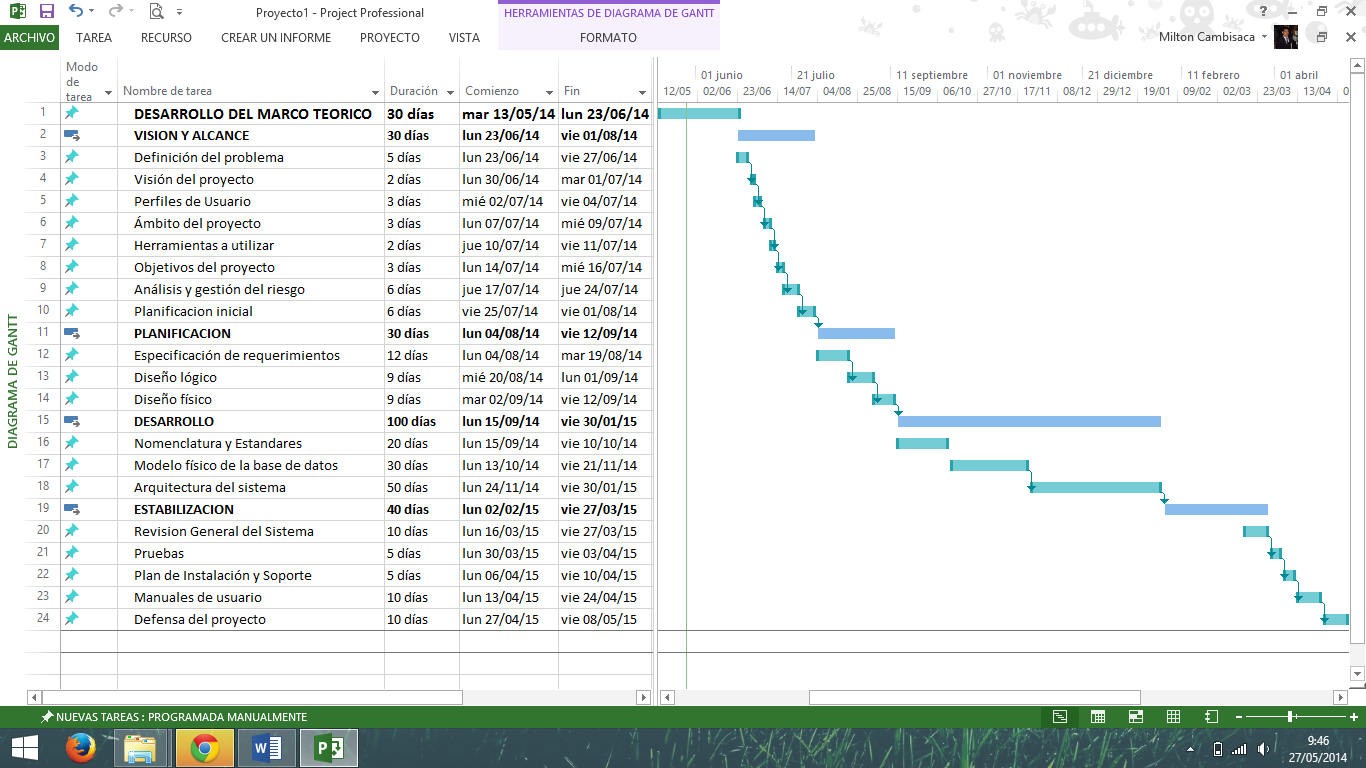
Posibilita la observación de cada una de las actividades planteadas y de esta manera verificar el cumplimiento de las mismas, en los tiempos establecidos.

*Diagrama Gantt*

Las figuras 17-4 y 18-4 presentan el desarrollo del proyecto de manera cronológica en cada una de sus fases según la metodología de desarrollo MSF.

**Figura 17-4.** Diagrama de Gantt

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Diagrama de Gantt de seguimiento*

**Figura 18-4.** Diagrama de Gantt en Seguimiento

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Estudio de Factibilidad.

##### *Factibilidad Técnica.*

*Software Mínimo Requerido*

Implantación de un Servidor en PostgreSQL.

Implantación de un Servidor Web en Ngnx.

Sistema operativo Ubuntu

*Hardware Requerido*

1 SERVIDOR

Procesador: Intel Xeon 2.0 Ghz

Memoria RAM: 4GB

Disco duro: 500 GB

*Herramientas Utilizadas*

Sublime Text

Suite Ofimática 2010

Vertavelo

PostgresSQL

##### *Factibilidad Operativa.*

1. Es necesario que las interfaces gráficas sean intuitivas y amigables posibilitando el correcto funcionamiento del sistema, además de que el personal a cargo de su utilización esté debidamente capacitado
2. El sistema debe ser administrado y mantenido por personal que posea conocimientos en el área de sistemas informáticos, juntamente con un pequeño grado de veterinaria, que puede irse adquiriendo mediante la administración del sistema.

##### *Factibilidad Legal.*

El sistema no interfiere, tampoco incumple ninguna norma ni reglamento interno dentro del Ministerio de Agricultura, Ganadería, Acuacultura y Pesca.

##### *Factibilidad Económica.*

La Tabla 32-4, detalla el costo del proyecto, tomando en cuenta cada uno de las necesidades que inciden en el desarrollo del mismo.

**Tabla 32-4:** Costo Proyecto

|  |  |  |  |
| --- | --- | --- | --- |
| **HARDWARE** | | | |
| **Cantidad** | **Descripción** | **Precio U.** | **Precio Total** |
| 1 | Infraestructura de la Red | 350,00 | 350,00 |
| 2 | Pc Portátiles | 750,00 | 1500,00 |
| **RECURSO HUMANO** | | | |
| **Cantidad** | **Descripción** | **Precio U.** | **Precio Total** |
| **2** | Desarrollador por 1 año | 15000,00 | 30000,00 |
| **OTROS GASTOS** | | | |
| **Cantidad** | **Descripción** | **Precio U.** | **Precio Total** |
| 1 | Servicio Telefónico | 200,00 | 200,00 |
| 1 | Servicio de Internet | 400,00 | 400,00 |
| Continúa | | | |
| 1 | Servicio de Transporte | 200,00 | 200,00 |
| 1 | Insumos de Oficina | 250,00 | 250,00 |
| 1 | Servicio de electricidad | 180,00 | 180,00 |
| **Costo Total del Proyecto** | | | **33080,00** |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Cada uno de los precios NO incluye I.V.A,** se agregará respectivamente cuando la factura sea emitida.

### Planificación

#### Definición y Descripción de Requerimientos.

*Objetivo del SRS*

La especificación de requerimientos de software (SRS), tiene como principal objetivo brindar una referencia sobre las funciones que el sistema va a realizar, tomando en cuenta los alcances y límites del proyecto.

*Ámbito*

El Ministerio de Agricultura, Ganadería, Acuacultura y Pesca respetando su misión ha creado el proyecto llamado “Botiquín Veterinario” que buscar impulsar el desarrollo de agrícolas y ganaderos de la provincia de Morona Santiago, por lo que es necesario evaluar resultados estadísticos para determinar la efectividad del proyecto.

El proyecto requiere que en los cantones de la provincia se conformen asociaciones, a los cuales se le hará entrega de paquetes de medicinas y utensilios veterinarios, para emprender un negocio y beneficiar a agricultores y ganaderos del sector.

*Sistema de Control de Botiquines SICOBO*

**SICOBO** tiene la función de registrar las ventas y comprar de cada una de las asociaciones mencionadas anteriormente que estará a cargo de un funcionario del MAGAP, para posteriormente procesar los datos y entregar resultados que permitan evaluar la eficiencia del proyecto y dar seguimiento a cada una de las asociaciones inscritas en el programa.

*SRS Visión General*

El documento de especificación de requerimientos esta realizado para definir a detalle cada uno de los requerimientos y funcionalidades que debe cumplir el producto final, y describir la arquitectura, organización y flujo de datos interno. Una vez concluido el trabajo de este documento, es expuesto al cliente para su análisis y aprobación.

*SRS Descripción General*

Paga facilitar agilizar y reducir tiempos en el proceso de registro de compras y ventas por parte de las asociaciones, se ha desarrollado SICOBO, que permite el ingreso de datos por parte de los técnicos encargados de asociaciones y en su posterior se puedan emitir reportes estadísticos para su análisis.

*Razones que motivan la realización del Proyecto*

* Apoyar a los ganaderos y agrícolas a mejorar su servicio.
* Facilitar el registro de datos referentes a las actividades económicas de las asociaciones.
* Determinar la efectividad del programa de Botiquín Veterinario

*Perspectiva del Producto*

El proyecto en un plan piloto que se va a ser ejecutado en la provincia de Morona Santiago, sin embargo una vez que pase por una fase de pruebas puede ser extendido al resto del país para brindar beneficio a muchos sectores más.

*Funciones del Producto*

Funciones principales del sistema:

* Registro de Ventas de asociaciones.
* Registro de compras de asociaciones
* Registro de inventario de asociaciones
* Generación de reportes estadísticos.

*Características del Usuario*

SICOBO posee la siguiente jerarquía de usuarios.

* Administrador.
* Funcionario encargado.
* Técnicos.

*Limitaciones Generales*

* El sistema informático no registra ni emite facturas ni otros comprobantes obligatorios del SRI.
* Al no estar presente directamente en las asociaciones el sistema no valida si las transacciones de compras y ventas sean correctas.

*Supuestos y Dependencias*

Causas que pudieran afectar la culminación de los requerimientos del sistema:

* Cambio de requerimientos una vez el proyecto ya esté en desarrollo.
* La información ingresada por parte de los encargados debe ser correcta.

SICOBO, a continuación los requerimientos que deben ser implementados durante su desarrollo:

##### *Interfaces externas*

*Interfaz de usuarios*

Los usuarios del sistema tendrán a su disposición interfaces gráficas para el ingreso, modificación y eliminación según sus permisos lo permitan. Se usan también notificaciones para determinar errores y procesos exitosos por parte del usuario. Las interfaces de usuario son diseñadas con la ayuda de un Framework especializado en lograr la mejor experiencia de usuario. La tabla 33-4 y 34-4, presentan los requisitos mínimos de Hardware para el servidor y el cliente respectivamente.

**Tabla 33-4:** Requisitos Servidor

|  |  |
| --- | --- |
| **Característica** | **Requerimiento Mínimo** |
| Procesador | Intel Xeon 2.0 Ghz |
| Memoria RAM | 8 GB |
| Disco duro | 1TB |
| Interfaz de Red | 100 Mbps |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 34-4:** Requisitos Cliente

|  |  |
| --- | --- |
| **Característica** | **Requerimiento Mínimo** |
| Computador | 1.8 GHz, 1GB RAM, 500GB DE DISCO. |
| Acceso a internet | 512kbps |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Impresora*

Necesaria para la presentación de información impresa al usuario que ofrece el sistema mediante reportes.

##### *Interfaces de Software.*

SICOBO requiere de la siguiente plataforma que se detalla en la tabla 35-4, para el correcto funcionamiento del servidor, esto debido a las bondades que ofrece.

**Tabla 35-4:** Interfaces de Software

|  |  |
| --- | --- |
| **Tipo** | **Plataforma de software** |
| Sistema operativo | Ubuntu Linux |
| Servidor Web | Ngnx |
| DBMS | PostgreSQL |
| Plataforma de Programación | Python |
| Plataforma de aplicaciones | Mozilla Firefox, Google Chrome, Internet Explorer. |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En el caso del cliente:

*Navegador*

Este sistema esta implementado en internet por lo cual el usuario solo necesita un computador con conexión a internet y un navegador como Mozilla Firefox, Google Chrome, Internet Explorer (en su última versión).

##### *Interfaz de comunicación.*

* La plataforma donde se asienta el sistema es un servidor web Ngnx.
* Es necesario un server que puede proveer la institución o ser contratado en la nube.

##### *Requerimientos Funcionales SICOBO.*

A continuación se detallan cada uno de los requerimientos funcionales del sistema SICOBO, definiendo datos de entrada, procesos y resultados.

1. *Administración de asociaciones:* SICOBO debe permitir el registro, edición y eliminación de datos de las asociaciones que ingresen al programa de botiquines veterinarios.
2. *Registro de ventas:* El sistema debe permitir a los técnicos encargados de las asociaciones registrar sus ventas teniendo en cuenta que una venta contiene datos del cliente, fecha, detalles. Cada producto que se registre en una venta debe también especificar para que especie de animal y que enfermedad va a tratarse.
3. *Registro de compras:* El sistema debe permitir a los técnicos encargados registrar las compras que las asociaciones realizan para reponer su mercadería.
4. *Administración de inventario:*El inventario debe actualizarse automáticamente cada vez que se registre compras o ventas, sin embargo, debe registrarse movimientos en el inventario cuando haya ingresado o salido un producto que no sea por compra o venta. Ej: devoluciones. Cada ingreso o salida de mercadería debe generar un registro en un kardex.
5. *Gestión de productos caducados:*Sabiendo que los productos tienen tiempo de caducidad el sistema debe notificar al usuario cuando productos del inventario estén por caducar para que las asociaciones tomen las medidas pertinentes.
6. *Generación de reportes:*El sistema debe permitir al técnico responsable generar reportes estadísticos que indiquen la actividad de las asociaciones, detallando sus ganancias mensuales, así mismo la cantidad de personas a quienes prestan servicios, y los porcentajes de animales y enfermedades que tratan.

*Requerimiento 1: Administración de asociaciones:*

*Descripción:*SICOBO debe permitir el registro, edición y eliminación de datos de las asociaciones que ingresen al programa de botiquines veterinarios.

*Entradas*

* Nombre
* Responsable
* Coordenadas de Geo localización
* Ubicación
* Técnico encargado
* Observaciones

*Procesos*

* El usuario administrador crea, modifica o elimina registros de asociaciones.
* El administrador asignar asociaciones a usuarios técnicos para que estén a su cargo.

*Salidas*

* Confirmación: Fallido o exitoso.

*Requerimiento 2: Registro de ventas*

*Descripción:*En el sistema los técnicos encargados de las asociaciones deben poder registrar sus ventas teniendo en cuenta que una venta contiene datos del cliente, fecha, detalles. Cada producto que se registre en una venta debe también especificar para que especie de animal y que enfermedad va a tratarse.

*Entradas*

* Cliente
* Fecha
* Productos
* Enfermedades a tratar con la venta
* Especies a tratar con la venta

*Procesos*

* El técnico encargado registra las ventas
* El técnico encargado registra los detalles de la venta
* El técnico registra las especies animales y enfermedades a tratar con los productos de la venta.

*Salidas*

* Confirmación: Fallido o exitoso

*Requerimiento 3: Registro de compras*

*Descripción:*En el sistema los técnicos encargados deben poder registrar las compras que las asociaciones realizan para reponer su mercadería.

*Entradas*

* Fecha de compra
* Detalles de productos comprados

*Procesos*

* El técnico registra las compras.
* El técnico registra el detalle de los productos comprados

*Salidas*

* Confirmación: Fallido o exitoso.

*Requerimiento 4: Administración de inventario*

*Descripción:*El inventario debe actualizarse automáticamente cada vez que se registre compras o ventas, sin embargo, debe registrarse movimientos en el inventario cuando haya ingresado o salido un producto que no sea por compra o venta. Ejemplo: devoluciones. Cada ingreso o salida de mercadería debe generar un registro en un kardex.

*Entradas*

* Cantidad
* Costo unitario
* Producto

*Procesos*

* Cada vez que se registre una compra o venta el sistema automáticamente actualiza el inventario.
* El usuario técnico registra salidas y entradas del inventario cuando no sea por motivo de compra o venta.
* Cada vez que se registra un ingreso o salida el sistema registra el moviendo en un kardex.

*Salidas*

* Confirmación: Fallido o exitoso.

*Requerimiento 5: Gestión de productos caducados.*

*Descripción:*Sabiendo que los productos tienen tiempo de caducidad el sistema debe notificar al usuario cuando productos del inventario estén por caducar para que las asociaciones tomen las medidas pertinentes.

*Entradas*

* Cantidad de productos del mismo tipo y con misma fecha de caducidad
* Fecha de caducidad

*Procesos*

* El sistema verifica si los productos del inventario esta caducados o próximos a caducarse tomando en cuenta su fecha de caducidad.
* El sistema notifica a los usuarios.

*Salidas*

* Notificaciones: Caducado o próximo a caducarse.

*Requerimiento 6: Reportes:*

*Descripción:*El sistema debe permitir al técnico responsable generar reportes estadísticos que indiquen la actividad de las asociaciones, detallando sus ganancias mensuales, así mismo la cantidad de personas a quienes prestan servicios, y los porcentajes de animales y enfermedades que tratan.

*Entradas*

* Solicitud de vista de reportes.

*Procesos*

* El usuario solicita al sistema la generación de un reporte.
* El usuario selecciones el tipo de reporte a visualizar
* El sistema despliega el reporte solicitado.

*Salidas*

* Despliegue: reporte solicitado por el usuario.

##### *Requerimientos No Funcionales.*

La siguiente lista detalla un grupo de requerimientos no funcionales es decir que no forman parte dentro del sistema que son necesarios para que este funcione de la mejor manera.

1. Seguridad
2. Mantenibilidad
3. Fiabilidad
4. Disponibilidad
5. Escalabilidad.
6. Facilidad de uso.

##### *Actores.*

Estos son los actores que interactúan con el sistema:

* Administrador.
* Directivo
* Técnicos

##### *Casos De Uso.*

En la figura 19-4 y tabla 36-4 se muestra el caso de uso del requerimiento # 1, que trata sobre la administración de asociaciones.



**Figura 19-4.** Caso de Uso. Requerimiento # 1

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 36-4:** Caso de Uso. Requerimiento # 1

|  |  |
| --- | --- |
| **Número:** Requerimiento 1 | **Usuario:** Administrador |
| **Nombre historia:** Administración de asociaciones | |
| **Programador responsable:** Jorge Peña, Milton Cambisaca | |
| **Descripción:** SICOBO debe permitir el registro, edición y eliminación de datos de las asociaciones que ingresen al programa de botiquines veterinarios. | |
| **Curso típico de eventos:** | |
| **ACCIONES DE ACTORES** | **RESPUESTA DELSISTEMA** |
| * El administrador solicita realizar operaciones de registro edición o eliminación. | * Despliega la interfaz gráfica adecuada para la acción solicitada por el usuario. |
| * El administrador registra, edita o elimina datos de asociaciones. | * Se valida cada una de las entradas y se ejecuta la acción. |
| **Observaciones:**  La información que se ingresa al sistema debe ser correcta, caso contrario la aplicación no procede a registrar los datos. | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 20-4 y tabla 37-4 se muestra el caso de uso del requerimiento # 2, que trata sobre el registro de Ventas



**Figura 20-4.** Caso de Uso. Requerimiento # 2

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 37-4:** Caso de Uso. Requerimiento # 2

|  |  |
| --- | --- |
| **Número:**  Requerimiento 2 | **Usuario:** Técnico. |
| **Nombre historia:** Registro de ventas. | |
| **Programador responsable:** Jorge Peña, Milton Cambisaca | |
| **Descripción:** En el sistema los técnicos encargados de las asociaciones deben poder registrar sus ventas teniendo en cuenta que una venta contiene datos del cliente, fecha, detalles. Cada producto que se registre en una venta debe también especificar para que especie de animal y que enfermedad va a tratarse. | |
|  | |
| **Curso Típico de Eventos:** | |
| **ACCIONES DE ACTORES** | **RESPUESTAS DEL SISTEMA** |
| 1.   El usuario solicita registrar ventas de asociaciones | 2.   Despliega la interfaz gráfica adecuada para la acción solicitada por el usuario. |
| 3.   El usuario ingresa los datos necesarios | 1.   Antes de que la información sea almacenada en la base de datos, el sistema valida las entradas y envía la información al servidor. |
| **Observaciones:** | |
| La información que se ingresa al sistema debe ser correcta, caso contrario la aplicación no procede a realizar la acción. | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 21-4 y tabla 38-4 se muestra el caso de uso del requerimiento # 3, que trata sobre el registro de Compras



**Figura 21-4.** Caso de Uso. Requerimiento # 3

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 38-4:** Caso de Uso. Requerimiento # 3

|  |  |
| --- | --- |
| **Número:** Requerimiento 3 | **Usuario:** Técnico |
| **Nombre historia:** Registro de compras | |
| **Programador responsable:** Jorge Peña, Milton Cambisaca | |
| **Descripción:** En el sistema los técnicos encargados deben poder registrar las compras que las asociaciones realizan para reponer su mercadería. | |
| **Curso típico de eventos:** | |
| **ACCIONES DE ACTORES** | **RESPUESTA DELSISTEMA** |
| * El usuario solicita el registro de compras. | * Despliega la interfaz gráfica adecuada para la acción solicitada por el usuario. |
| * El usuario ingresa datos necesarios | * El sistema valida las entradas y envía datos al servidor para su almacenamiento en la base de datos. |
| **Observaciones:**  Todos los datos deben estar ingresados correctamente, caso contrario el sistema notifica al usuario y no ejecuta la acción. | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 22-4 y tabla 39-4 se muestra el caso de uso del requerimiento # 4, que trata sobre la administración de Inventario



**Figura 22-4.** Caso de Uso. Requerimiento # 4

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 39-4:** Caso de Uso. Requerimiento # 4

|  |  |  |
| --- | --- | --- |
| **Número:**  Requerimiento 4 | **Usuario:** Técnico | |
| **Nombre historia:** Administración de inventario | | |
| **Programador responsable:** Jorge Peña, Milton Cambisaca | | |
| **Descripción:** El inventario debe actualizarse automáticamente cada vez que se registre compras o ventas, sin embargo, debe registrarse movimientos en el inventario cuando haya ingresado o salido un producto que no sea por compra o venta. Ej: devoluciones. Cada ingreso o salida de mercadería debe generar un registro en un kardex. | | |
| Continua | | |
| **Curso Típico de Eventos:** | | |
| **ACCIÓN DE ACTORES** | | **RESPUESTAS DEL SISTEMA** |
| 1.   El usuario solicita el registrar entradas o salidas de inventario | | 2.   Despliega la interfaz gráfica adecuada para la acción solicitada por el usuario. |
| 3.  El usuario ingresa los datos necesarios. | | 4.   El sistema valida las entradas, actualiza en inventario y genera un registro en el kardex. |
| **Observaciones:** | | |
| Si la información ingresada no es correcta el sistema notifica al usuario y detiene la acción. | | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 23-4 y tabla 40-4 se muestra el caso de uso del requerimiento # 5, que trata sobre la gestión de productos caducados.



**Figura 23-4.** Caso de Uso. Requerimiento # 5

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 40-4:** Caso de Uso. Requerimiento # 5

|  |  |
| --- | --- |
| **Número:** Requerimiento 5 | **Usuario:** Técnico |
| **Nombre historia:** Gestión de productos caducados. | |
| **Programador responsable:** Jorge Peña, Milton Cambisaca. | |
| **Descripción:** Sabiendo que los productos tienen tiempo de caducidad el sistema debe notificar al usuario cuando productos del inventario estén por caducar para que las asociaciones tomen las medidas pertinentes. | |
| **Curso típico de eventos:** | |
| **ACCIONES DE ACTORES** | **RESPUESTA DELSISTEMA** |
| * El usuario solicita verificar productos caducados | * El sistema obtiene fechas de productos y compara para encontrar productos caducados, notifica al usuario. |
| **Observaciones:**  El sistema compara con las fecha actual de sistema del cliente, se debe tener precaución que la fecha del computador sea la correcta. | |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 24-4 y tabla 41-4 se muestra el caso de uso del requerimiento # 6, que trata sobre la generación de reportes.



**Figura 24-4.** Caso de Uso. Requerimiento # 6

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Tabla 41-4:** Caso de Uso. Requerimiento # 6

|  |  |
| --- | --- |
| **Número:**  Requerimiento 6 | **Usuario:** Administrativo |
| **Nombre historia:** Generación de reportes | |
| **Programador responsable:** Jorge Peña, Milton Cambisaca | |
| **Descripción:** La aplicación debe permitir al técnico responsable generar reportes estadísticos que indiquen la actividad de las asociaciones, detallando sus ganancias mensuales, así mismo la cantidad de personas a quienes prestan servicios, y los porcentajes de animales y enfermedades que tratan. | |
| Continua | |
| **Curso Típico de Eventos:** | |
| **ACCIONES DE ACTORES** | **RESPUESTAS DEL SISTEMA** |
| 1.   El usuario solicita generar reportes. | 2.   El sistema despliega en pantalla los reportes disponibles. |
| 3.  El usuario selecciona el reporte a visualizar. | 4.   El sistema genera y despliega en pantalla el reporte solicitado. |
| **Observaciones:** | |

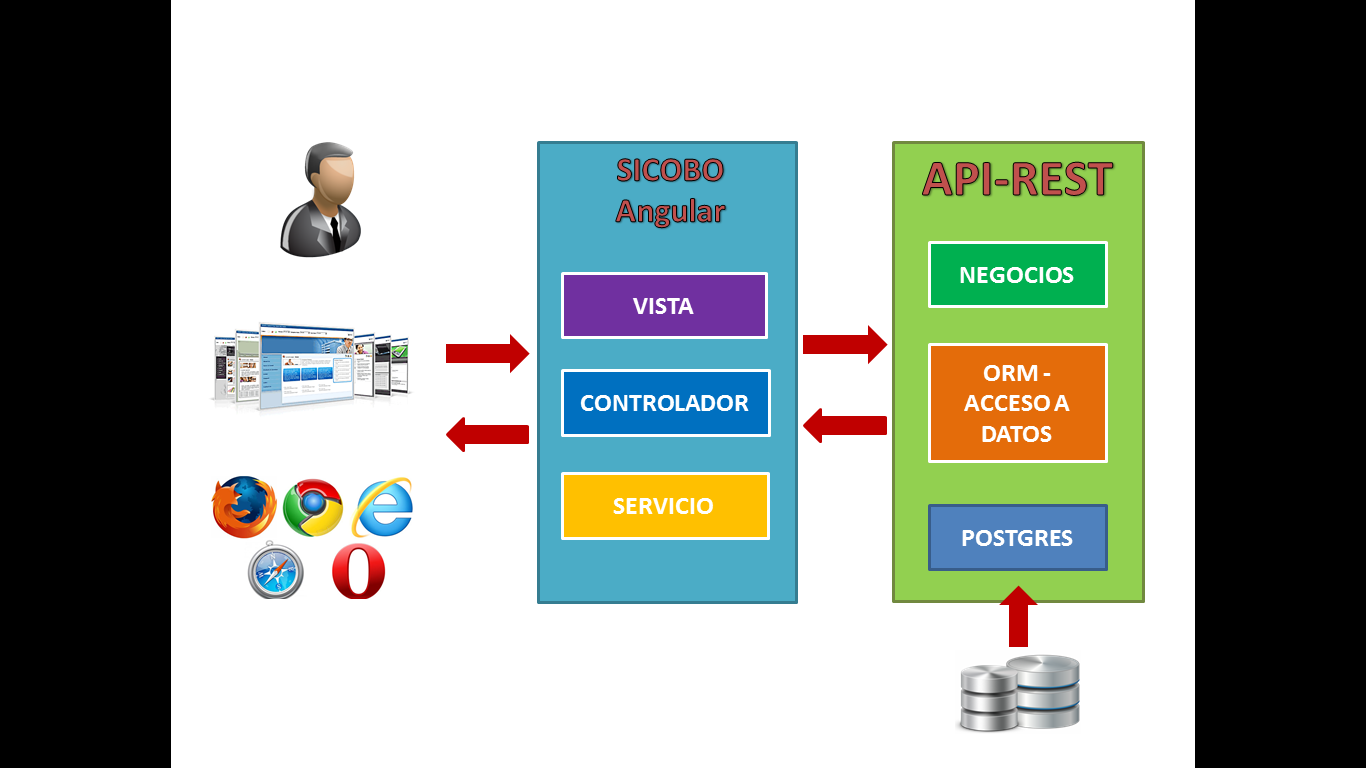
**Fuente:**  PEÑA, J., CAMBISACA, M., 2015



##### *Arquitectura apropiada para la solución.*

*Arquitectura Interna de SICOBO.*

Las figuras 25-4 y 26-4 muestran la arquitectura del sistema.

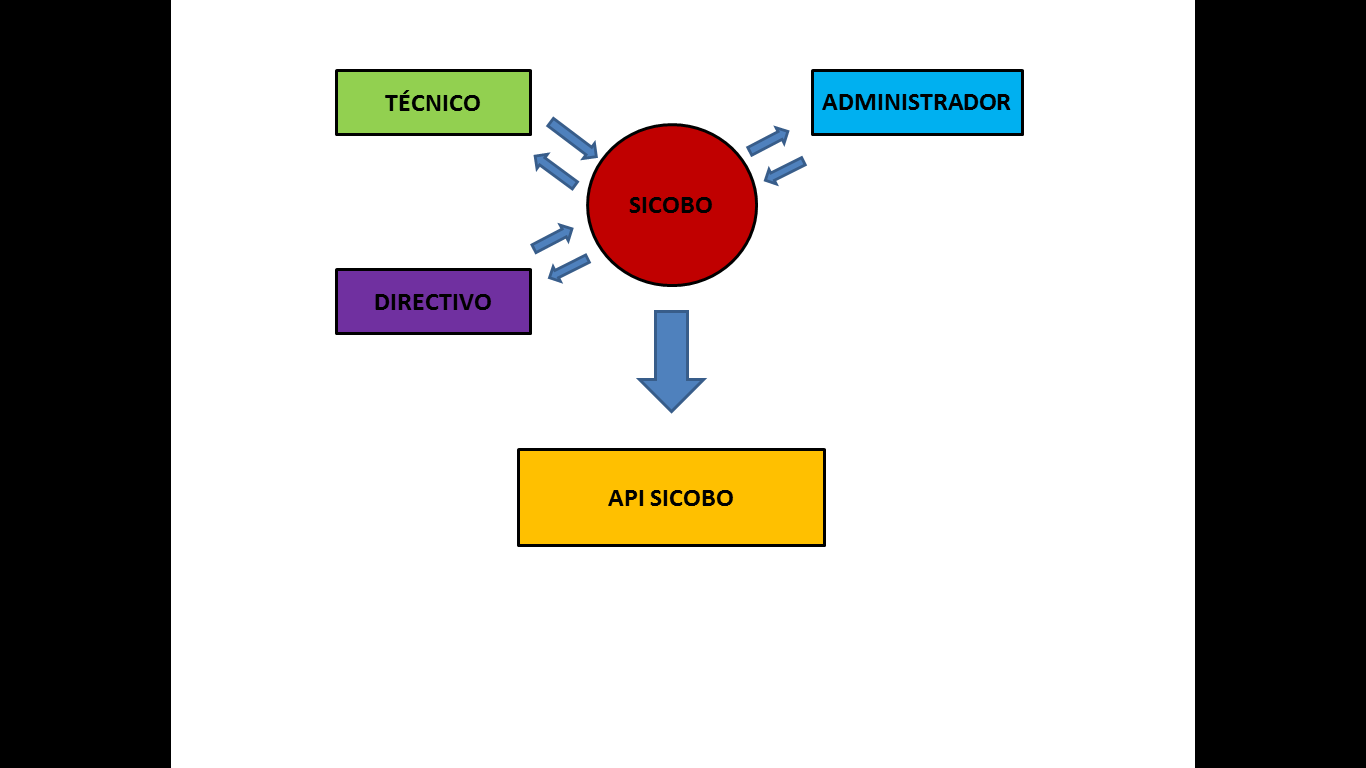


**Figura 25-4.** Arquitectura Interna

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Descripción de la Arquitectura del Sistema*

El siguiente gráfico representa una vista más amplia y general sobre la arquitectura del sistema.



**Flujo de datos**

**Figura 26-4.** Arquitectura Externa

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Diseño Lógico.

##### *Diagramas de Secuencia.*

En la figura 27-4 se muestra cómo funciona el sistema en el requerimiento # 1 según el diagrama de secuencia sobre la Administración de asociaciones.



**Figura 27-4.** Req.1 Diagrama de secuencia de Administración de asociaciones

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 28-4 se muestra cómo funciona el sistema en el requerimiento # 2 según el diagrama de secuencia sobre el Registro de ventas.



**Figura 28-4.** Req.2 Diagrama de secuencia de Registro de ventas.

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 29-4 se muestra cómo funciona el sistema en el requerimiento # 3 según el diagrama de secuencia sobre el Registro de compras.



**Figura 29-4.** Req.3 Diagrama de secuencia de Registro de compras

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 30-4 se muestra cómo funciona el sistema en el requerimiento # 4 según el diagrama de secuencia sobre la Administración de inventario.



**Figura 30-4.** Req.4 Diagrama de Secuencia de Administración de inventario

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 31-4 se muestra cómo funciona el sistema en el requerimiento # 5 según el diagrama de secuencia sobre la gestión de productos caducados.



**Figura 31-4.** Req.5 Diagrama de secuencia de gestión de productos caducados

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

En la figura 32-4 se muestra cómo funciona el sistema en el requerimiento # 6 según el diagrama de secuencia sobre la generación de reportes.

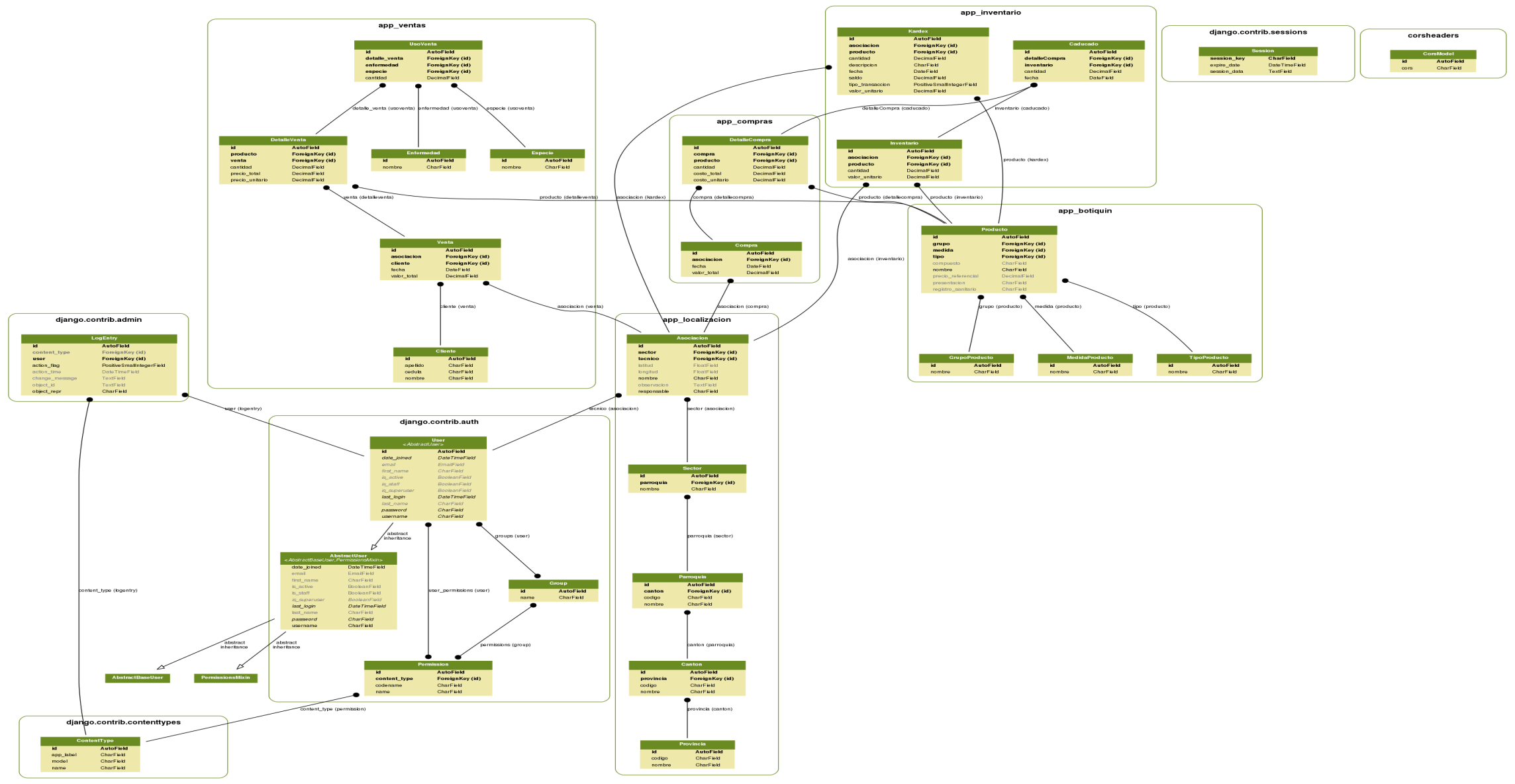


**Figura 32-4.** Req.6 Diagrama de Secuencia de generación de reportes.

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *Diagrama de Clases.*

Como se muestra en la figura 33-4 se detalla los atributos de cada una de las entidades del sistema SICOBO.

****

**Figura 33-4.** Diagrama de Clases.

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

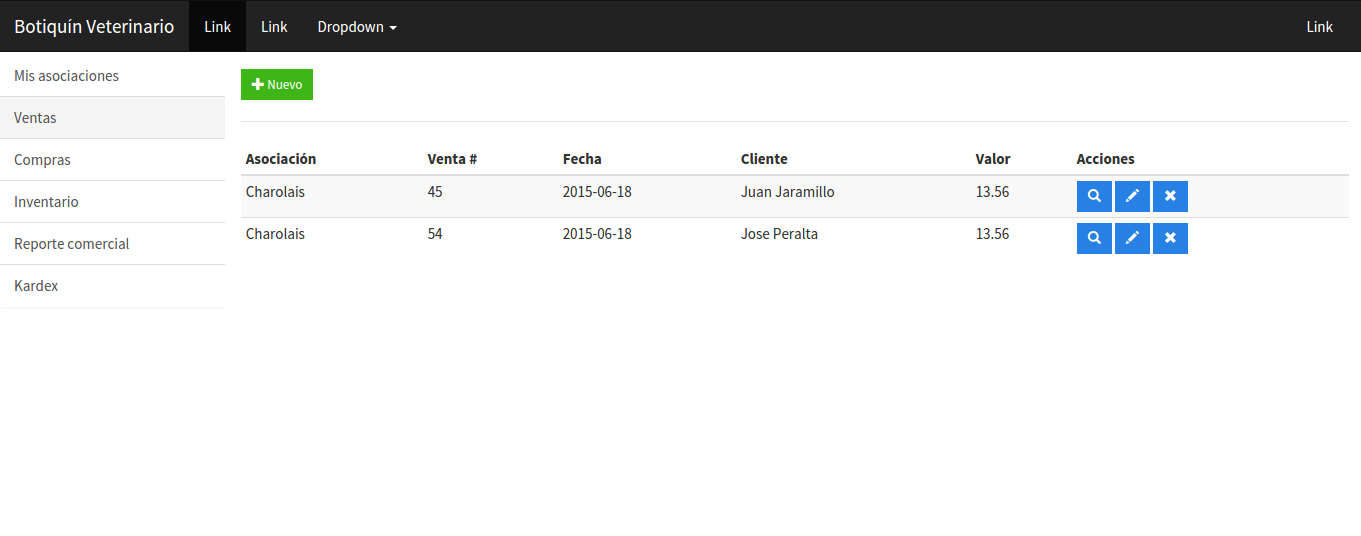
##### *Diseño de interfaces de usuario.*

La forma en que el usuario interactúa con el sistema se lo realiza mediante interfaces gráficas en un navegador de internet.

Para el proceso de validación de información el usuario visualizará mensajes de error o confirmación al ejecutar ciertas acciones que enviarán la información a la base de datos.

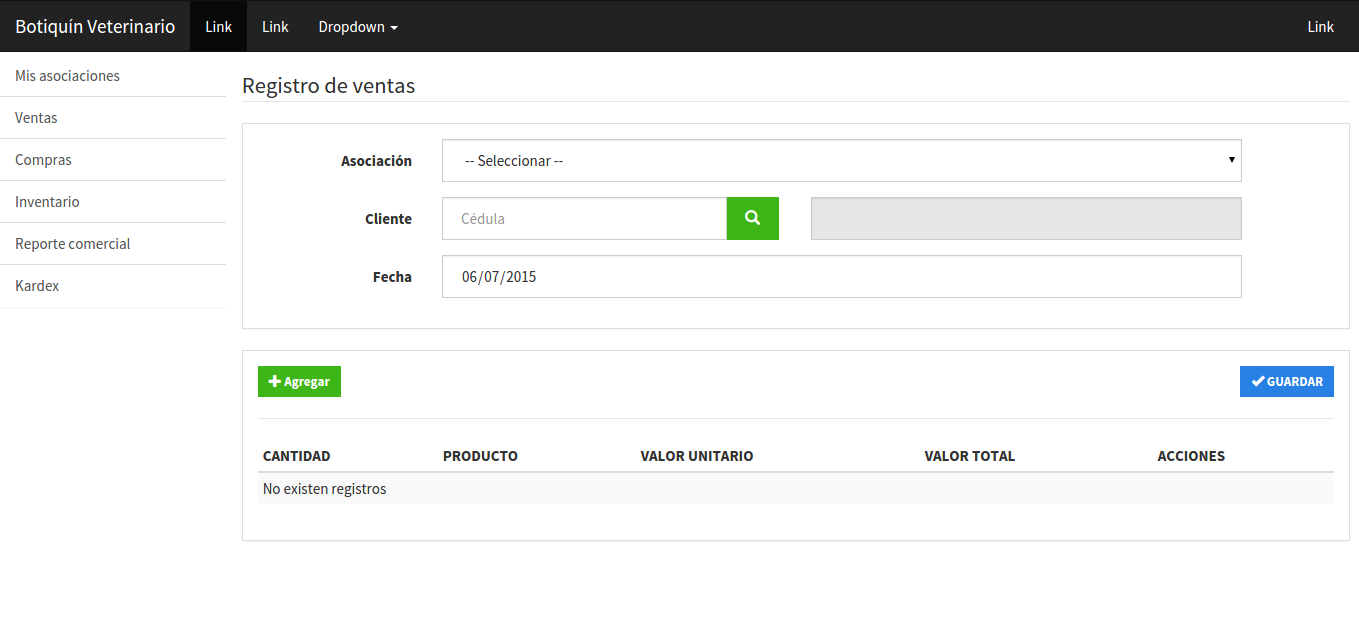
Las figuras 34-4 hasta la 39-4 muestran un prototipo de las interfaces gráficas de usuario.

*Registro de ventas.*

**

**Figura 34-4.** Registro de ventas

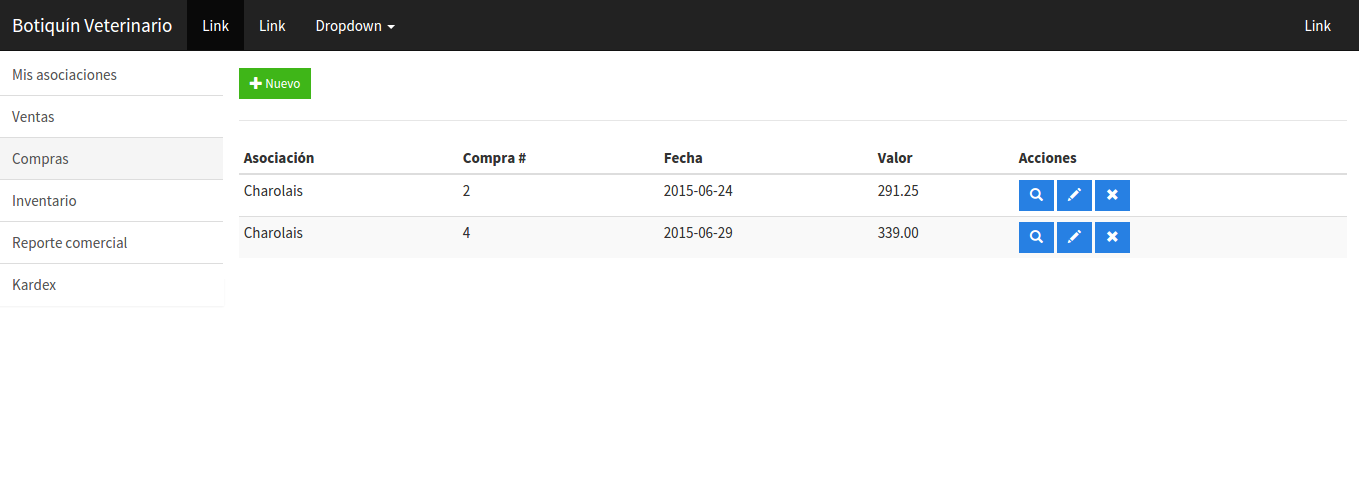
**Fuente:**  PEÑA, J., CAMBISACA, M., 2015



**Figura 35-4.** Registro de ventas

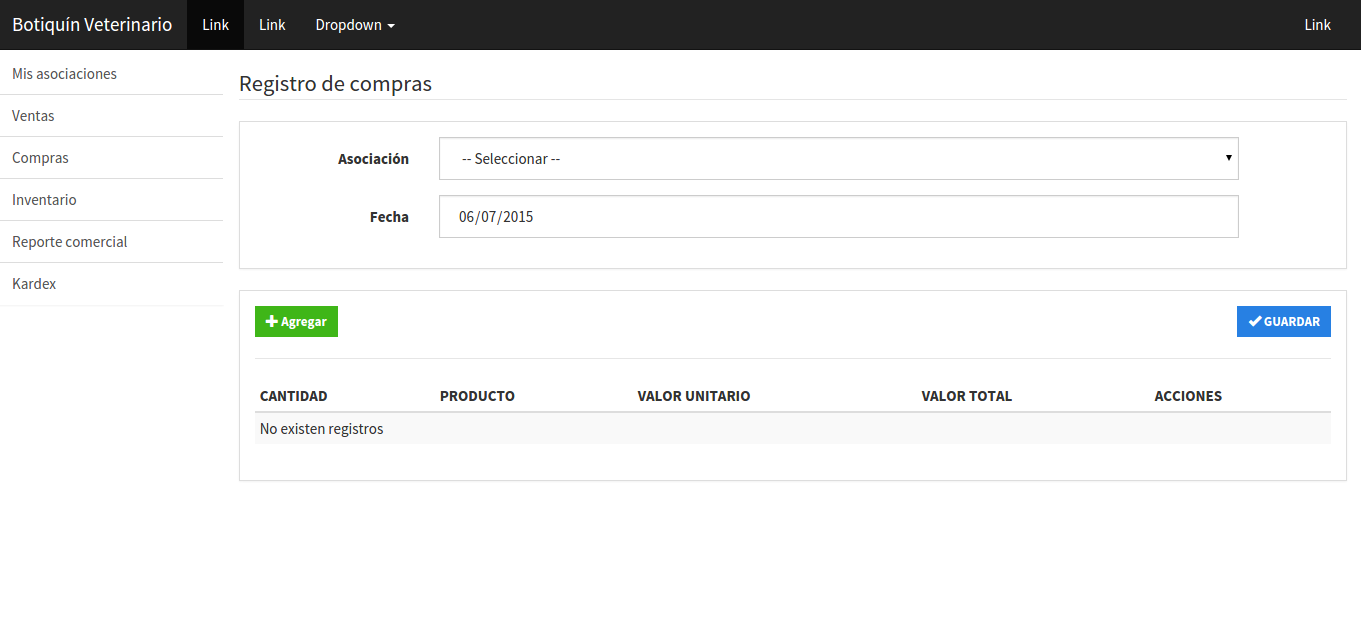
**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Registro de compras*

****

**Figura 36-4.** Registro de compras

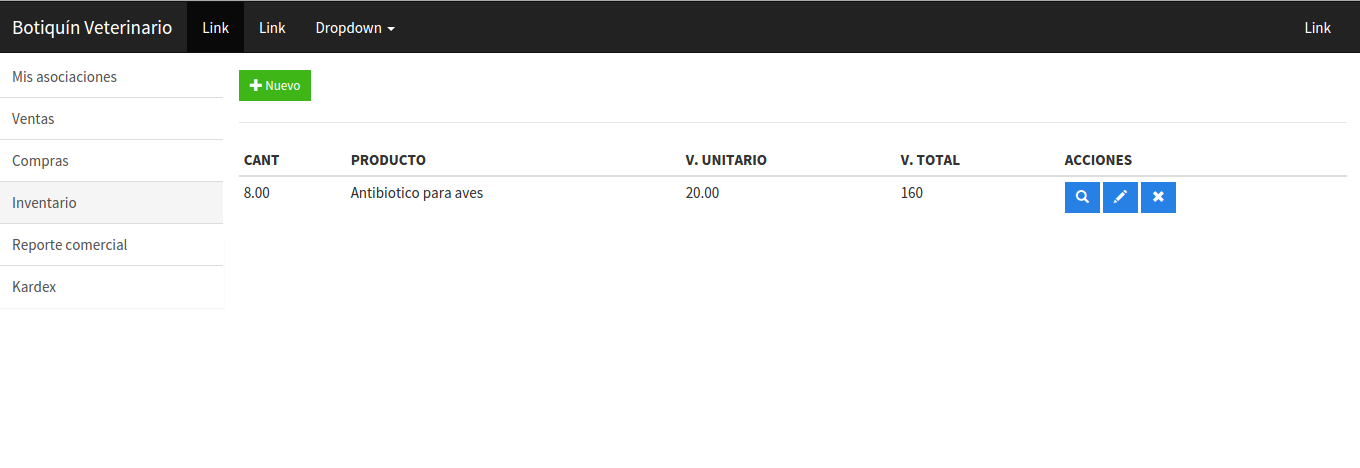
**Fuente:**  PEÑA, J., CAMBISACA, M., 2015



**Figura 37-4.** Registro de compras

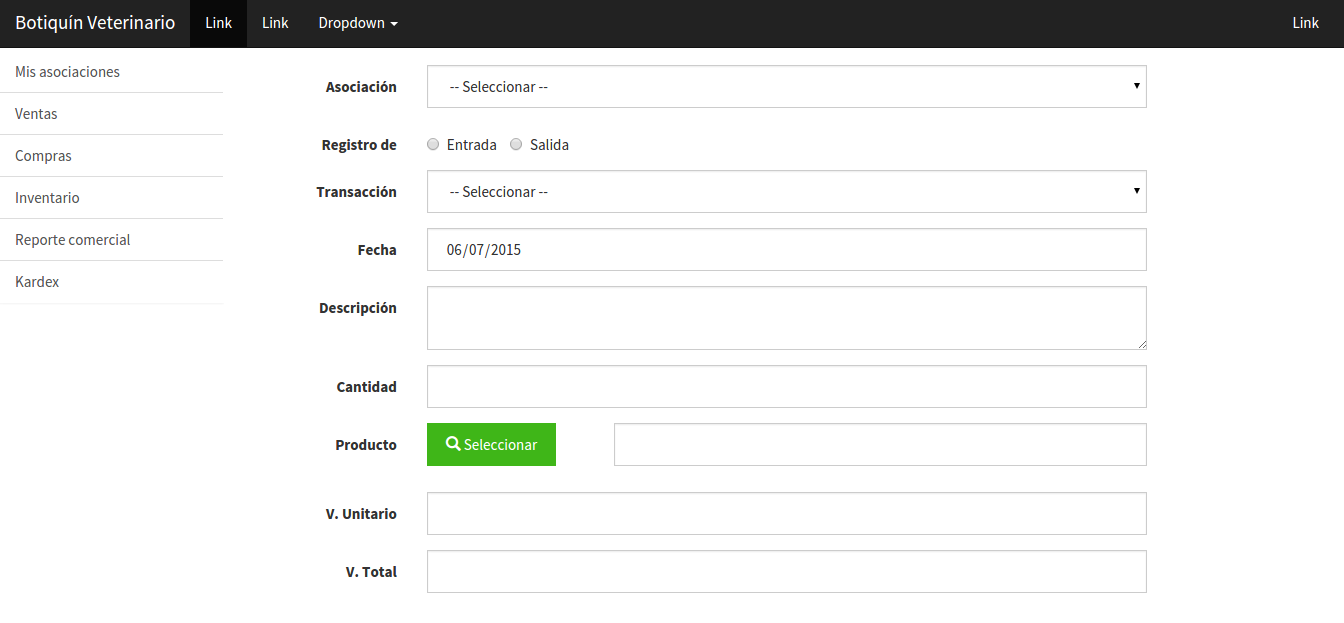
**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Inventarios*

****

**Figura 38-4.** Inventarios

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

****

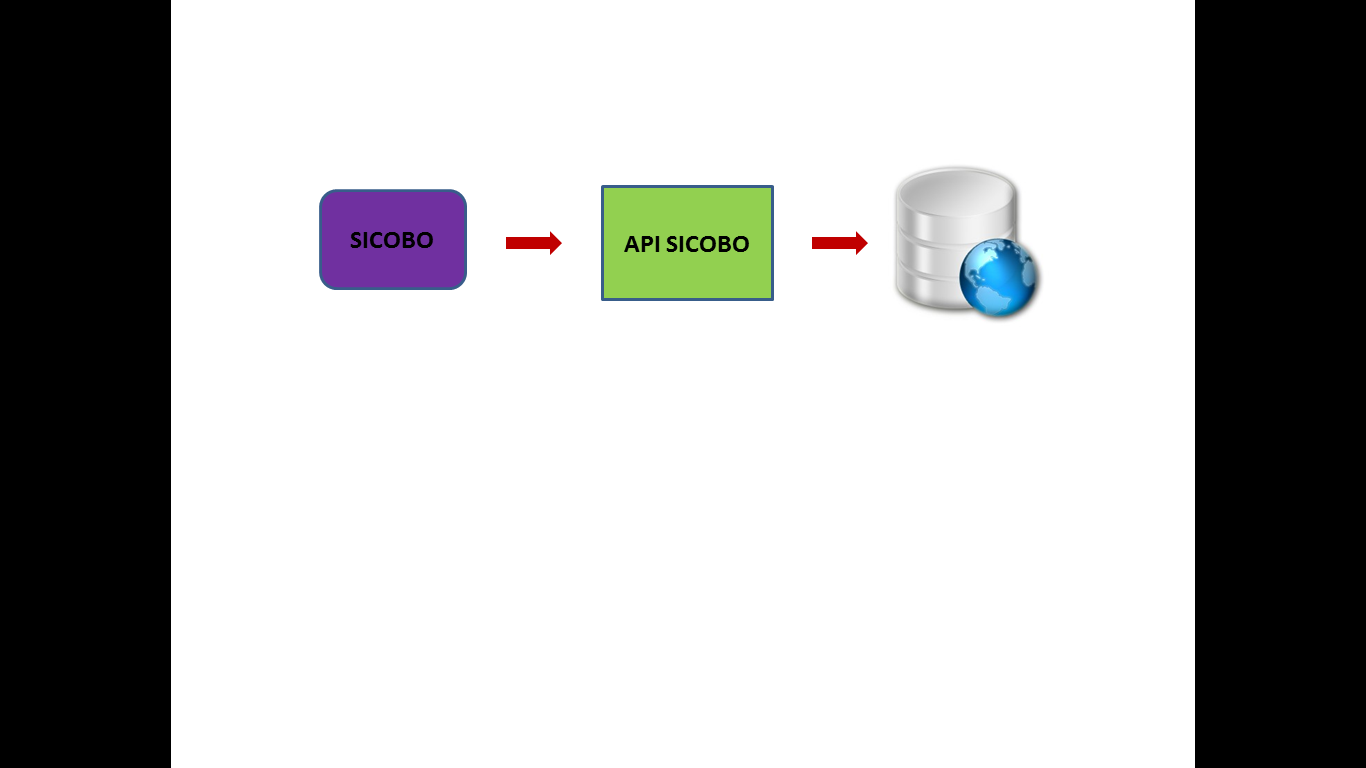
**Figura 39-4.** Inventarios

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### Diseño Físico.

##### *Diagrama de Implementación.*

La figura 40-4 muestra el diagrama de implementación del sistema SICOBO, en el que indica que las conexiones a la base de datos se realizan a través del API REST.

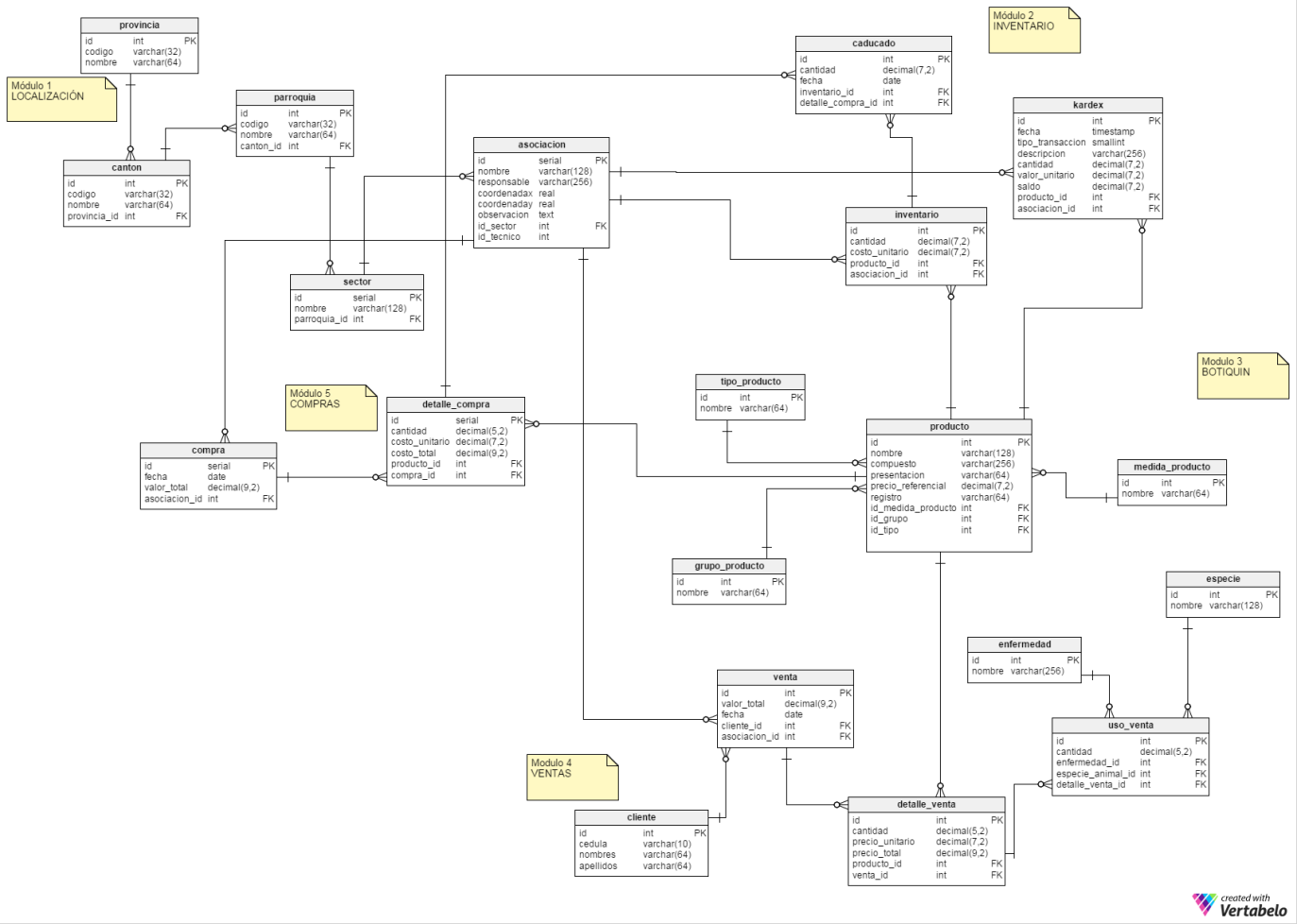


**Figura 40-4.** Diagrama de Implementación

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

* + - 1. *Modelo Físico de Base de Datos.*

La siguiente figura 41-4 muestra el modelo físico de la base de datos del Sistema de Control de Botiquín (SICOBO), en la que se observa las tablas con sus relaciones, juntamente con cada uno de sus atributos y tipos de valor.



**Figura 41-4.** Modelo Físico BD

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

### Desarrollo



#### Estándar de diseño de base de datos.

##### *Nombre de los objetos de una base de datos.*

Para identificar a los objetos en la Base de Datos, es necesario manejar una nomenclatura adecuada.

Ejemplo:

* nombre\_objeto

El nombre del objeto será escrito en minúsculas. En caso de ser palabras compuestas, de la misma manera deberán estar todas las palabras en minúsculas y separadas por el signo “\_”.

Ejemplo:

* botiquin\_producto
* cliente

##### *Sentencias SQL.*

Cada una de las sentencias SQL a utilizar para las consultas y las vistas, deben estar en mayúsculas.

Ejemplo:

DELETE FROM tabla WHERE tabla\_id = 001

#### Estándar de Programación de Aplicaciones.

##### *Extensiones de archivos.*

La tabla 42-4 describe las principales extensiones de los formatos que se utilizan en la aplicación, dando a conocer como deberían ser nombrados, para ser utilizados y facilitar su detección.

**Tabla 42-4:** Archivos con sus extensiones

|  |  |  |
| --- | --- | --- |
| **Tipo de Archivo** | **Extensión** | **Nomenclatura** |
| Python | .py | Nombre-archivo.py |
| Imágenes | .gif / .jpg / .png | imagen.gif  imagen.jpg  imagen.png |
| Hojas de Estilo | .css | nombre-archivo.css |
| JavaScript | .js | nombre-archivo.js |
| Página HTML | .html | nombre-archivo.html |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *Convenciones De Nombres.*

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. Estos pueden dar información sobre la función de un identificador.

*Clases*

Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).

Ejemplo:

Class Regla;

Class ReglaCurso;

*Métodos*

Deben ser verbos, cuando son compuestos estarán separados por el signo “\_”.

Ejemplo:

ingresar\_sistema()

cerrar\_sesion()

*Variables*

Las variables de clase o método junto con las instancias se escriben en minúsculas y si son compuestas deberán estar separadas por el signo “\_” a excepción de las constantes. Los nombres que se asignan a las variables deben contener significado y en lo posible ser cortos, además debe utilizarse mnemónicos al momento de asignar el nombre a la variable, esto para indicar a un observador casual su función.

No es recomendable utilizar variables que contengan un solo carácter, excepto cuando se emplea en nombres para variables temporales como, x, y, z, a.



#### ***Api REST***.

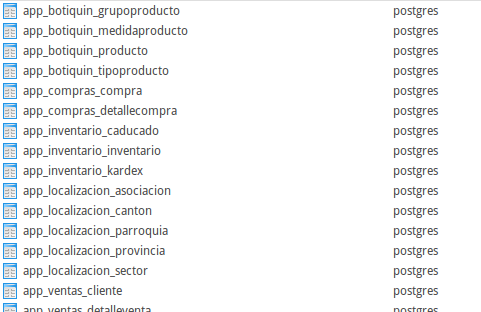
En este apartado se explica cómo está desarrollado el api-REST que envía y recibe los datos desde y hacia las aplicaciones que la requieran.

Toda la aplicación está desarrollada en el lenguaje de programación Python y con los frameworks Django y django-rest-framework.

##### *Capa de datos*

En la capa de datos reposan los datos que son necesarios para que el sistema funcione correctamente, puede estar formada por uno o más gestores de bases de datos. PostgreSQL es el que se utiliza en este caso, motor de bases de datos que permite trabajar con herramientas para almacenar, modificar y extraer información.

La figura 42-4 indica las tablas que forman parte del sistema SICOBO



**Figura 42-4.** Tablas de la base de datos

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *ORM - Django.*

Django posee una herramienta muy poderosa que comunica nuestra aplicación con el repositorio de datos, es decir, es la capa de acceso a datos de nuestro sistema. Esta herramienta es conocida como ORM – Object Relational Mapping, el cual está preparado para soportar diferentes tipos de bases de datos entre ellas Postgres.

##### *Capa de negocios.*

En esta capa residen lo modelos, una representación de datos a nivel de aplicación para poder procesarlos y obtener resultados para nuevamente ser almacenados en la base de datos. Esta capa está dividida en módulos los cuales se menciona a continuación:

* Localización, Ventas, Compras, Inventario, Botiquín

Las figura 43-4 muestra en el editor de textos sublime-text una implementación de esta capa en el módulo de ventas.



**Figura 43-4.** Negocios – módulo de ventas

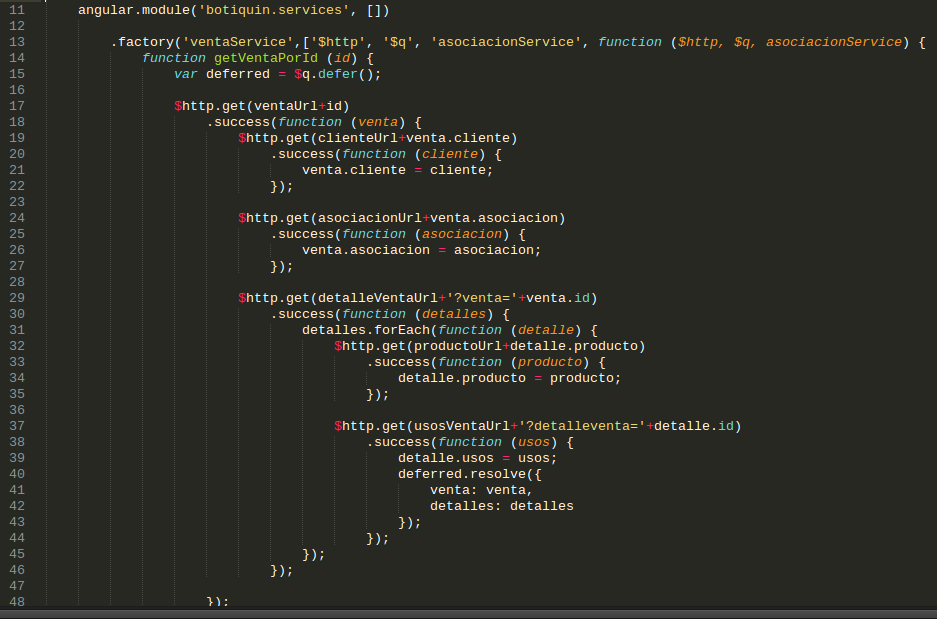
**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

#### SICOBO - AngularJS.

Las nuevas tecnologías nos han dado gran facilidad al hacer aplicaciones distribuidas, anteriormente se podía encontrar únicamente aplicaciones donde toda la lógica estaba en el servidor. Actualmente se puede crear aplicaciones donde la lógica se puede distribuir entre el servidor y el cliente, en el desarrollo de esta aplicación se hace uso del Framework Angular que ayuda a implementar el patrón de diseño MVC en el lado del cliente.

##### *Capa de servicios*

La capa de servicios brinda las funcionalidades para conectar la aplicación con el proveedor de datos, en este caso, es api-rest que se menciona anteriormente. La figura 44-4 muestra la manera en que esta implementada la capa de servicios de esta aplicación.



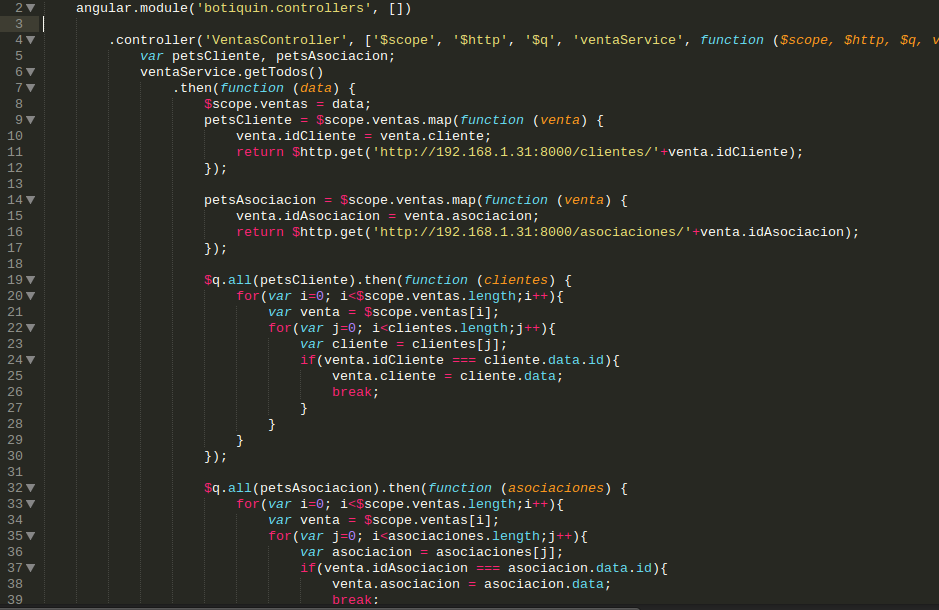
**Figura 44-4.** Capa de servicios

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *Capa de controladores*

La capa de controladores permite manipular la forma en que los datos serán presentados al usuario, actúa de intermediario para tratar los datos que recolecta del cliente para procesarlos y enviarlos al repositorio de datos.

La figura 45-4 muestra la manera en que esta implementada la capa de controladores en la aplicación.



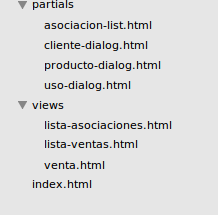
**Figura 45-4.** Capa de controladores

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

##### *Capa de Vistas.*

Conformada por todos los documentos HTML de la aplicación como muestra la figura 46-4, en estos se utilizan las directivas y demás componentes que se encuentran definidos en el modelo de la información.

Archivos de la capa de vista, permiten mostrar la interfaz gráfica de la aplicación con la que interactúa el usuario.



**Figura 46-4.** Capa de Vistas

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

### Estabilización

#### Revisión del Sistema.

**SICOBO** ha sido desarrollado en base a una arquitectura de sistemas distribuidos, por un lado tenemos un APIREST que provee toda la funcionalidad para el envío y recepción de datos, y por otro lado tenemos una aplicación del lado del cliente implementa el patrón de diseño MVC.

– Modelo - Vista - Controlador.

* Centraliza los aspectos de seguridad y transacciones, que serán responsabilidad del API.
* Mejora en el rendimiento y potencia de la aplicación web en el lado del cliente al implementar lógica de negocio.
* División del sistema en pequeños componentes reutilizables.

*Código fuente*

*SICOBO APIREST*

* *Modelo*

class Producto(models.Model):

class Meta:

verbose\_name = 'producto'

verbose\_name\_plural = 'productos'

nombre = models.CharField(max\_length = 128)

compuesto = models.CharField(max\_length = 256, null = True, blank=True)

presentacion = models.CharField(max\_length = 64, null = True, blank=True)

precio\_referencial = models.DecimalField(max\_digits = 7, decimal\_places = 2, null = True, blank=True)

registro\_sanitario = models.CharField(max\_length = 64, null = True, blank=True)

medida = models.ForeignKey(MedidaProducto)

tipo = models.ForeignKey(TipoProducto)

grupo = models.ForeignKey(GrupoProducto)

def edit(self):

return '<span class="icon-pencil"></a>'

def \_\_unicode\_\_(self):

return self.nombre

edit.allow\_tags = True

* *ViewSet*

class ProductoViewSet(viewsets.ReadOnlyModelViewSet):

queryset = Producto.objects.all()

serializer\_class = ProductoSerializer

def get\_queryset(self):

queryset = Producto.objects.all()

id\_medida = self.request.QUERY\_PARAMS.get('medida',None)

id\_tipo = self.request.QUERY\_PARAMS.get('tipo', None)

id\_grupo = self.request.QUERY\_PARAMS.get('grupo',None)

keyword = self.request.QUERY\_PARAMS.get('keyword', None)

if id\_medida is not None and id\_tipo is not None and id\_grupo is not None:

queryset = queryset.filter(medida\_\_id=id\_medida, tipo\_\_id=id\_tipo, grupo\_\_id=id\_grupo)

elif id\_medida is not None and id\_tipo is not None:

queryset = queryset.filter(medida\_\_id=id\_medida, tipo\_\_id=id\_tipo)

elif id\_medida is not None and id\_grupo is not None:

queryset = queryset.filter(medida\_\_id=id\_medida, grupo\_\_id=id\_grupo)

elif id\_tipo is not None and id\_grupo is not None:

queryset = queryset.filter(tipo\_\_id=id\_tipo, grupo\_\_id=id\_grupo)

elif id\_medida is not None :

queryset = queryset.filter(medida\_\_id=id\_medida)

elif id\_tipo is not None:

queryset = queryset.filter(tipo\_\_id=id\_tipo)

elif id\_grupo is not None:

queryset = queryset.filter(grupo\_\_id=id\_grupo)

elif keyword is not None:

queryset = queryset.filter(Q(nombre\_\_icontains=keyword)|Q(compuesto\_\_icontains=keyword)

|Q(presentacion\_\_icontains=keyword)|Q(precio\_referencial\_\_icontains=keyword)|Q(registro\_sanitario\_\_icontains=keyword))

return queryset

* *Serializer*

class ProductoSerializer(serializers.HyperlinkedModelSerializer):

class Meta:

model = Producto

fields = ('id', 'nombre','compuesto','presentacion','precio\_referencial','registro\_sanitario','medida','tipo','grupo')

*SICOBO AngularJS*

* *Controlador*

.controller('ProductoController', ['$scope', '$modalInstance', 'productoService', function ($scope, $modalInstance, productoService) {

var dataList = {};

$scope.productos = [];

$scope.error = false;

$scope.mensajeError = '';

$scope.empty = true;

var contains = function (str, searchString) {

return str.toLowerCase().indexOf(searchString.toLowerCase()) > -1;

};

var startsWith = function (str, searchString) {

return str.toLowerCase().indexOf(searchString.toLowerCase()) === 0;

};

var filter = function (){

$scope.empty = true;

$scope.productos = dataList.filter(function (obj) {

if(contains(obj.nombre, $scope.keyword) || contains(obj.compuesto, $scope.keyword)

|| contains(obj.presentacion, $scope.keyword) || contains(obj.registro\_sanitario, $scope.keyword)) {

return obj;

}

});

if($scope.productos.length > 0){

$scope.empty = false;

}

}

productoService.getTodos()

.then(function (data) {

dataList = data;

$scope.productos = data;

if(data.length>0){

$scope.empty = false;

}

});

$scope.filtrarProductos = function (event) {

if(!event){

filter();

}else if(event.keyCode === 13){

filter();

}

};

$scope.seleccionarProducto = function (producto) {

$scope.productoSeleccionado = producto;

}

$scope.ok = function () {

if(!$scope.productoSeleccionado){

$scope.error = true;

$scope.mensajeError = 'No ha seleccionado ningun item.';

return;

}

$modalInstance.close($scope.productoSeleccionado);

}

$scope.cancelar = function () {

$modalInstance.dismiss('cancel');

}

}])

* *Servicio*

.factory('productoService', ['$http', '$q', function ($http, $q) {

function getProductosTodos () {

var deferred = $q.defer();

$http.get(productoUrl)

.success(function (data) {

deferred.resolve(data);

});

return deferred.promise;

}

return {

getTodos: getProductosTodos,

}

}])

* *Vista*

<div class="modal-header">

<button ng-click="cancelar()" type="button" class="close" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

<h4 class="modal-title" id="myModalLabel">Búsqueda de productos</h4>

</div>

<div class="modal-body">

<div class="form-inline">

<div class="form-group">

<label class="sr-only">Palabra clave</label>

<input ng-keyup="filtrarProductos($event)" ng-model="keyword" type="text" class="form-control" placeholder="Palabra clave">

</div>

<button ng-click="filtrarProductos()" class="btn btn-success">Buscar</button>

</div>

<hr>

<div ng-show="error" class="alert alert-danger" role="alert">

<strong>Error!</strong> {{mensajeError}}

</div>

<div class="table-responsive">

<table class="table table-striped table-condensed table-hover">

<thead>

<tr>

<th>NOMBRE</th>

<th>COMPUESTO</th>

<th>PRESENTACION</th>

<th>PRECIO REFERENCIAL</th>

<th>REGISTRO SANITARIO</th>

</tr>

</thead>

<tbody>

<tr>

<td ng-show="isEmpty" colspan="3">No existen registros</td>

</tr>

<tr ng-repeat="producto in productos" ng-class="{success: productoSeleccionado === producto}" ng-click="seleccionarProducto(producto)">

<td>{{producto.nombre}}</td>

<td>{{producto.compuesto}}</td>

<td>{{producto.presentacion}}</td>

<td>{{producto.precio\_referencial}}</td>

<td>{{producto.registro\_sanitario}}</td>

</tr>

</tbody>

</table>

</div>

</div>

<div class="modal-footer">

<button ng-click="cancelar()" type="button" class="btn btn-default">Cancelar</button>

<button ng-click="ok()" type="button" class="btn btn-primary" data-dismiss="modal">Aceptar</button>

</div>

#### Plan de Pruebas.

Antes de pasar a la fase de producción con un sistema informático se deben realizar pruebas para determinar si se ha cumplido con las expectativas del cliente y corregir cualquier posible error que aun pueda surgir.

En este proceso se ha solicitado la participación del personal del MAGAP:

Ing. Miguel Guzman – Administrativo botiquines

Ing. Miguel Guzman – Técnico botiquines

*Módulos a Evaluar*

La tabla 43-4 detalla los módulos que están sujetos a evaluación, además de los usuarios que harán pruebas en el sistema.

**Tabla 43-4:** Módulos A Evaluar

|  |  |  |
| --- | --- | --- |
| **Módulo** | **Acciones** | **Usuario evaluador** |
| API | Manejo de sistema administrativo | Administrativo |
| Ventas | Registro, edición y eliminación de ventas. | Técnico |
| Compras | Registro, edición y eliminación de compras. | Técnico |
| Inventario | Registro, edición de inventario y kardex. | Técnico |
| Reportes | Generación de reportes. | Administrativo |

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

*Entorno de Pruebas*

Para llevar a cabo un plan de pruebas se debe definir claramente cuál será la configuración del entorno:

*Servidor*

* Servidor virtual corriendo sobre VMware Workstation 11.
* Procesador Intel Core i7 2.40 GHz
* Ram 2GB
* Disco duro 50GB
* Sistema Operativo Linux Ubuntu
* Servidor Web Nginx

*Clientes*

* Procesador Intel Dual Core 2.0 Ghz
* Ram 2GB
* Disco Duro 500GB
* Sistema Operativo Windows
* Navegador Web Mozilla Firefox y Google Chrome

Las siguientes tareas deben ejecutarse sin ningún problema, esto para determinar que el sistema ha superado las pruebas realizadas.

* *API:* el usuario administrativo debe hacer uso de la interfaz administrativa para ingresar, editar y eliminar registros sin tener mayores contratiempos.
* *Módulo de ventas:* los usuarios técnicos deber realizar acciones de registro, edición y eliminación de datos correspondientes a ventas que realizan las asociaciones.
* *Módulo de compras:* los usuarios técnicos deber realizar acciones de registro, edición y eliminación de datos correspondientes a compras que realizan las asociaciones.
* *Inventarios:* los usuarios técnicos deber realizar acciones de registro, edición de datos correspondientes al inventario de productos de las asociaciones.
* *Reportes:* el usuario administrativo de generar reportes requeridos y validar que los datos obtenidos se presenten de acuerdo a los criterios del personal del MAGAP.

### Instalación

#### Plan de Instalación y Soporte.

*Pre-requisitos*

La instalación de SICOBO requiere de un servidor con las siguientes características mínimas.

* Procesador: Intel Xeon 2.0 Ghz
* Memoria RAM: 4GB
* Disco duro: 500GB
* Conexión a internet
* Sistema operativo Ubuntu

***Instalación de Dependencias***

**PIP**: Herramienta para instalar y administrar paquetes para Python

**Sudo apt-get install python-pip**

**Nginx**: Servidor web (se usa como proxy inverso y para cachear los archivos estáticos ‘img, js, css’ ) Lo instalamos con:

**Sudo apt-get install nginx**

**Supervisor**: Aplicación para administrar y supervisar las aplicaciones, aunque sirve para mucho más. Se instala con:

**Sudo apt-get install supervisor**

**Virtualenv**: Sirve para crear un entorno virtual de ejecución personalizado para una aplicación. Se instala con:

**Sudo apt-get install python-virtualenv**

**Postgres:** Gestor de bases de datos. Se instala con los siguientes comandos:

**sudo apt-get install postgres-9.4**

**sudo apt-get install postgresql-client-9.4**

**sudo apt-get install postgresql-contrib-9.4**

**sudo apt-get install libpq-dev**

**sudo apt-get install postgresql-server-dev-9.4**

**Psycopg2:** Utilidad para comunicar Python a PostgreSQL.

**sudo apt-get update**

**sudo apt-get install libpq-dev python-dev**

**sudo apt-get install python-psycopg2**

***Creación del Entorno Virtual***

En primer lugar hay que situarse dentro del directorio que será usado para la puesta en producción.

Ejemplo:

**cd /var/www/**

Se procede a crear el entorno virtual

**virtualenv env-sicobo**

Copiamos la aplicación dentro de este directorio y se procede a activar el entorno creado con el siguiente comando

**source env-sicobo/bin/activate**

La terminal debe lucir así

**(env) usario@root**

Esto logrará que todo lo que se instale quede almacenado en el directorio **/var/www/env-sicobo** sin afectar los paquetes del sistema.

Paso siguiente, hay que ubicarse en el directorio de la aplicación

**cd sicobo**

Se procede a instalar la lista de dependencias de la aplicación, las mismas están especificadas en el archivo **requirements.txt**:

**pip install –r requirements.txt**

También podemos instalar paquetes de manera individual, por ejemplo, para instalar el conector de **postgresql**:

**pip install pyscopg2**

***Instalación y Configuración De Gunicorn***

Para instalarlo hacemos de la misma manera:

**pip install gunicorn**

Ahora se debe configurar, para ello vamos a crear un archivo llamado **gunicorn-deploy.py** en la raíz de nuestra aplicación, (aunque el nombre puede ser cualquiera) con el siguiente contenido:

bind = "127.0.0.1:8001" # dirección a donde accederá Nginx

logfile = "/var/www/logs/nombreApp/gunicorn.log" # dirección donde estarán los logs de la aplicación

workers = 1 # dependerá en medida de la carga de trabajo que tenga la aplicación, también depende del hardware con que se cuente

loglevel = 'info' # tipo de logging

***Configuración de Supervisor***

Ahora se debe configurar **supervisord**, para eso se genera el fichero de configuración con

**echo\_supervisord\_conf > /etc/supervisord.conf**

Se edita el archivo de configuración:

gedit /etc/supervisord.conf

Y se des comenta las siguientes líneas quitándole el ;  (punto y coma):

[unix\_http\_server]

file=/tmp/supervisor.sock

[supervisord]

logfile=/var/log/supervisord.log

logfile\_maxbytes=50MB

logfile\_backups=10

loglevel=debug

pidfile=/var/run/supervisord.pid

nodaemon=false

minfds=1024

minprocs=200

[rpcinterface:supervisor]

supervisor.rpcinterface\_factory = supervisor.rpcinterface:make\_main\_rpcinterface

[supervisorctl]

serverurl=unix:

[program:nombreApp]

command=/var/www/ENV-nombreApp/bin/django\_unicorn -c /var/www/ENV-nombreApp/nombreApp/gunicorn-deploy.py

directory=/var/www/ENV-nombreApp/nombreApp/

autostart=true

autorestart=true

user=nombreUser

redirect\_stderr=true

stdout\_logfile=/var/www/logs/nombreApp/supervisord.log

Ahora se crea el script para que supervisord inicie con el sistema, para eso crearemos el archivo:

**Gedit /etc/init.d/supervisord**

Y le agregamos el siguiente contenido:

# Supervisord auto-start

#

# description: Auto-starts supervisord

# processname: supervisord

# pidfile: /var/run/supervisord.pid

SUPERVISORD=/usr/local/bin/supervisord

SUPERVISORCTL=/usr/local/bin/supervisorctl

case $1 in

start)

echo -n "Iniciando supervisord:"

$SUPERVISORD

echo

;;

stop)

echo -n "Deteniendo supervisord:"

$SUPERVISORCTL shutdown

echo

;;

restart)

echo -n "Deteniendo supervisord:"

$SUPERVISORCTL shutdown

echo

echo -n "Iniciando supervisord: "

$SUPERVISORD

echo

;;

esac

Y ahora le damos permisos de ejecución al archivo para que pueda iniciar con el sistema:

**Sudo chmod +x /etc/init.d/supervisord**

Actualizamos los enlaces para arrancar el servicio:

**Sudo update-rc.d supervisord defaults**

Iniciamos el servicio:

**Sudo /etc/init.d/supervisord start**

***Configuracion de Nginx***

Este paso es bastante sencillo, se crea el siguiente archivo de configuración de **nginx** para nuestra aplicación:

**Gedit /etc/nginx/sites-enabled/sicobo**

Y se agrega el siguiente contenido

server {

listen 9001; # puerto donde quieren que escuche nginx

server\_name www.dominio.com; # o 192.168.0.100, dirección a la cual accederemos

access\_log /var/log/nginx/nombreApp.access.log; # donde tendremos el log de la aplicación

location / { # a donde llamara nginx al acceder a www.dominio.com/

proxy\_pass http://127.0.0.1:8001;

proxy\_set\_header Host $http\_host;

}

location /static/ { # a donde accedera nginx cuando ingresemos en www.dominio.com/static/

alias /var/www/ENV-nombreApp/nombreApp/staticfiles/;

}

}

Y se reinicia nginx con:

**service nginx restart**

***Configuración Django***

Vamos a modificar el archivo de configuración de django:

**gedit sicobo/settings.py**

Se cambia la línea que dice **DEBUG = True** por el valor **DEBUG = False**

Agregamos los parámetros de la DB:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2', 'NAME': 'nombreDB',

'USER': 'usuarioDB',

'PASSWORD': 'contrasenaDB',

'HOST': 'localhost', # o el que necesiten

'PORT': '', # o el que esten usando

}

}

Buscamos la linea **ALLOWED\_HOSTS=[]** y le agregamos el dominio o la dirección por medio del cual accederemos, quedando algo como **ALLOWED\_HOSTS=[‘www.dominio.com’]**

Configuramos el directorio para los archivos estáticos, buscando la línea que dice

**STATIC\_ROOT = ‘’**

Luego se cambia el valor, colocándole la ruta absoluta donde queremos que estén nuestros archivos estáticos, en este caso sería más o menos así

STATIC\_ROOT='/var/www/sicobo/statics/'

Ya casi por finalizar, ejecutamos el siguiente comando:

**Python manage.py collectstatic**

Esto nos creara una carpeta con el nombre ‘**statics** en la ruta que especificamos en el **settings.py** **‘**, es allí donde estarán todos nuestros archivos estáticos.

Y por último se reinicia supervisord para que tome los nuevos cambios:

**Supervisorctl restart sicobo**

# CONCLUSIONES

1. Mediante los resultados alcanzados descritos en el capítulo III se puede concluir que, al tratarse de rendimiento, el Framework AngularJS supera en un 18,06% a EmberJS, determinando de esta forma que la hipótesis ha sido comprobada.
2. La investigación de los Frameworks AngularJS y EmberJS, que implementan el patrón Modelo, Vista, Controlador (MVC), requirió el aprendizaje profundo de: instalación, implementación, funcionalidad y aplicación para el desarrollo de los prototipos y la aplicación final.
3. Los indicadores especificados para medir el rendimiento entre los Framework AngularJS y EmberJS, permitieron la verificación clara y concisa de cada uno de ellos, obteniendo como resultado que AngularJS alcanzó un 48,89%, mientras que EmberJS obtuvo un 30,83%, dando como resultado final que el Framework AngularJS es el más adecuado para aplicaciones web que requieran de un mejor rendimiento.
4. Una fase fundamental del uso de la metodología MSF para un buen diseño y desarrollo, es la fase de Análisis de Requerimientos ya que en base a esto se puede modelar la correcta funcionalidad que debe cumplir y satisfacer el sistema.
5. Se alcanzó cumplir satisfactoriamente con los requerimientos planteados del sistema desarrollado para el MAGAP de Morona Santiago, garantizando el tratamiento correcto de los insumos y productos agrícolas – veterinarios.

# RECOMENDACIONES

1. Al realizar un análisis comparativo entre Framework de desarrollo, se debe elegir cuidadosamente los Framework involucrados, los cuales deben tener funcionalidad y características semejantes para obtener resultados certeros sin inconvenientes.
2. El patrón de arquitectura de software Modelo, Vista, Controlador MVC puede ser usado e implementado para el desarrollo de cualquier proyecto de aplicación web, en especial aquellos que manejan una gran cantidad de datos y transacciones complejas.
3. Realizar un estudio previo al uso e implementación de cualquier Framework de desarrollo, para poder seleccionar el que mejor se acople a los requerimientos y necesidades de la aplicación web.
4. Para el desarrollo de aplicaciones web es importante que el desarrollador esté actualizando diariamente su conocimiento en nuevas mejoras o modificaciones que se le añadan al Framework de su elección.

**GLOSARIO**

**A**

**AngularJS**.- Es un framework JavaScript de desarrollo de aplicaciones web en el lado cliente, desarrollado por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página y utiliza el patrón MVC (Model-View-Controller).

**B**

**Backend**.- Todas las tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del navegador web, generalizándose más que nada en tres lenguajes, Html , CSS Y JavaScript.

**E**

**EmberJs**.- Es un framework que nos permite usar la metodología MVC en la realización de interfaces web complejas como las usadas en la mayoría de aplicaciones web actuales donde casi toda la interacción ocurre en el lado del cliente.

**F**

**Frameworks**.- Conjunto de herramientas que proporcionan es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, puede servir de base para la organización y desarrollo de software

**Frontend**.- Es la parte que procesa la entrada de datos que se efectuó desde el front-end. Son los procesos que utiliza el administrador del sitio con sus respectivos sistemas para resolver las peticiones de los usuarios.

**G**

**Github**.- Plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

**K**

**Kardex**.- Registro organizado de la mercancía que se tiene en un almacén.

**M**

**MVC**.- Model-View-Controller, es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación).

**R**

**Rendimiento**.- Proporción que surge entre los recursos empleados para obtener algo y el resultado que se consigue.

**S**

**Servidor.-** Es un ordenador remoto que provee los datos solicitados por parte de los navegadores de otras computadoras.

**Software**.- Son aplicaciones que ayudan a crear otras aplicaciones y/o son aplicaciones finales para ser usadas.

**W**

**Web service**.- Conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

**BIBLIOGRAFÍA**

**ALEGSA, Leandro.** Definición de Aplicación Web. alegsa.com.ar. Santa Fé-Argentina. 2015.

http://www.alegsa.com.ar/Dic/aplicacion%20web.php

2015- 06- 15

**ÁLVAREZ, Cecilio.** Introducción a Servicios REST. arquitecturajava.com. 2015

http://www.arquitecturajava.com/servicios-rest/

2015- 06- 17

**ÁLVAREZ, Miguel.** Binding en AngularJS y Doble Binding. desarrolloweb.com. 2014

http://www.desarrolloweb.com/articulos/binding-angularjs-doble-binding.html

2014- 10- 20

**ECUADOR, MINISTERIO DE AGRICULTURA, GANADERIA, ACUACULTURA Y PESCA.** Valores Misión Visión. agricultura.gob.ec.Quito-Ecuador. 2015.

http://www.agricultura.gob.ec/valores-mision-vision/

2015- 06- 19

**ESQUIVA, Alejandro.**  JSONI- ¿Qué es y para qué sirve JSON?. geekytheory.com. 2015

https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/

2015- 06- 1

**ESTADOS UNIDOS, GOOGLE.** ¿Qué es Angular?. docs.angularjs.org. 2015

https://docs.angularjs.org/guide/introduction

2015- 06- 18

**ESTADOS UNIDOS, GOOGLE.**  Preguntas más frecuentes. docs.angularjs.org. 2015

https://docs.angularjs.org/misc/faq

2015- 06- 18

**ESTADOS UNIDOS, GOOGLE.**  Conceptual Overview. docs.angularjs.org. 2015

https://docs.angularjs.org/guide/concepts

2015- 06- 19

**ESTADOS UNIDOS, GOOGLE.** Data Binding. docs.angularjs.org. 2015

https://docs.angularjs.org/guide/databinding

2015- 06- 19

**ESTADOS UNIDOS, GOOGLE.** Creating Custom Directives. docs.angularjs.org. 2015

https://docs.angularjs.org/guide/directive

2015- 06- 20

**FLANAGAN, David.**  JavaScript la Guía Definitiva. books.google.com.ec. 2015. p.15.

https://books.google.com.ec/books/about/JavaScript.html?id=mriFGgAACAAJ&redir\_esc=y

2015- 06- 15

**GONZÁLEZ, Lorenzo.** Servicios. cursoangularjs.es. Valencia-España. 2014

http://www.cursoangularjs.es/doku.php?id=unidades:03\_servicios:01\_servicios

2014- 08- 29

**GONZÁLEZ, Lorenzo.** Instalación de AngulaJS. cursoangularjs.es. Valencia-España. 2014

http://www.cursoangularjs.es/doku.php?id=unidades:01\_introduccion:02\_instalacion

2014- 09- 06

**MOHAMED, Salvador.** Esquema de Funcionamiento de Servicios Web. es.scribd.com. 2015. p.1

http://es.scribd.com/doc/65908579/Esquema-de-Funcionamiento-de-Servicios-Web#scribd

2015- 06- 15

**MOREIRA, Valentín.**  Aplicaciones Web. es.scribd.com. Madrid-España. 2015. p.2

http://es.scribd.com/doc/75239310/Aplicaciones-Web

2015- 06- 15

**NATIVIDAD, Luis.**  Módulos y Controladores en AngularJS. frontendlabs.io. 2015.

http://frontendlabs.io/2264--modulos-y-controladores-en-angularjs

2015- 01- 05

**NATIVIDAD, Luis.**  Aprendiendo Directivas en AngularJS. frontendlabs.io 2015.

http://frontendlabs.io/2287--aprendiendo-directivas-en-angularjs

2015- 04- 06

**NATIVIDAD, Luis.** Hablemos de AngularJS. frontendlabs.io. 2015.

http://frontendlabs.io/2152--hablemos-de-angularjs

2014- 12- 18

**PÉREZ, Damián.**  Qué es JavaScript. maestrosdelweb.com. 2015. p.1

http://www.maestrosdelweb.com/que-es-javascript/

2015- 05- 03

**RUIZ, Fernando.**  Expresiones en AngularJS. fr2dev.com. 2015

http://fr2dev.com/2015/03/27/expresiones-en-angularjs/

2015- 03- 27

**SESHADRI, Shyam., Green, Brad.**  AngularJS Up & Running. EE.UU, O’Reilly Media. 2014. Pp.3-11,11-11.

2014- 12- 15

**TAVARES, José.** Servicio Web. es.slideshare.net. Santo Domingo-Republica Dominicana. 2015. Pp.1-8

http://es.slideshare.net/Tancrelluberes/trabajo-de-la-segunda-semana

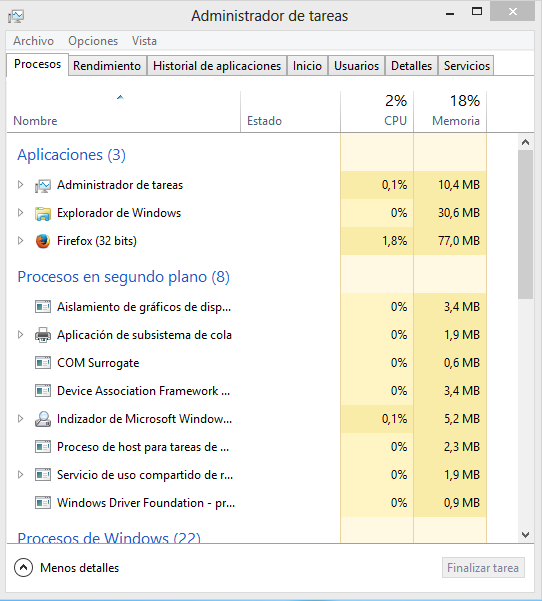
2015- 06- 16

**ANEXOS**

Anexo A: Resultados, pruebas con los Frameworks, AngularJS y EmberJS.

**Indicador CPU y Memoria RAM.**

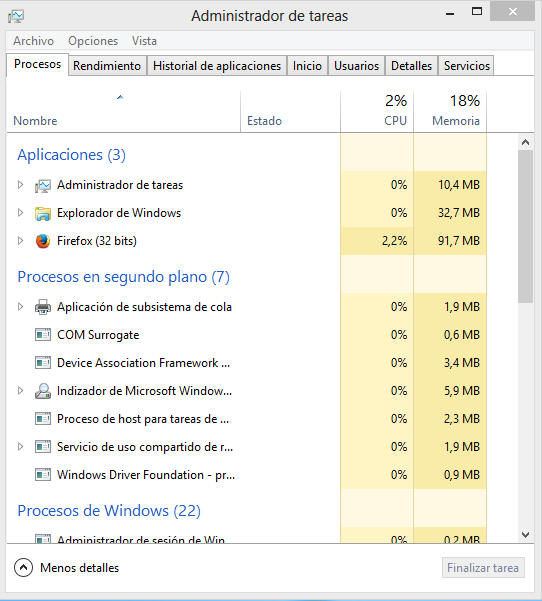
**AngularJS GET**

****

**Figura 47-4.** AngularJS GET, CPU y Memoria RAM

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

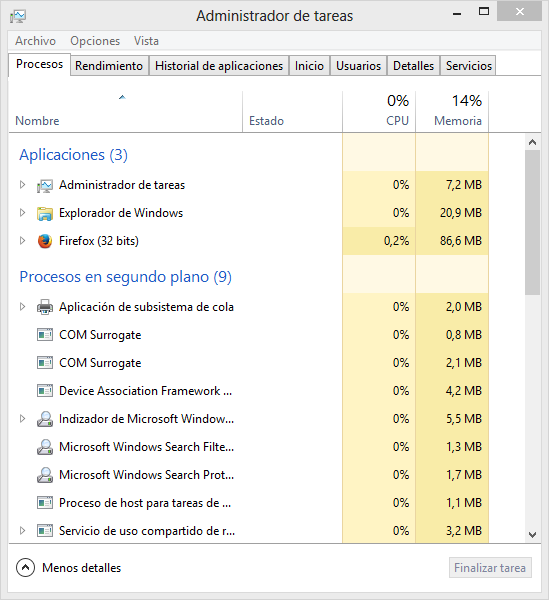
**EmberJS GET**

****

**Figura 48-4.** EmberJS GET, CPU y Memoria RAM

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

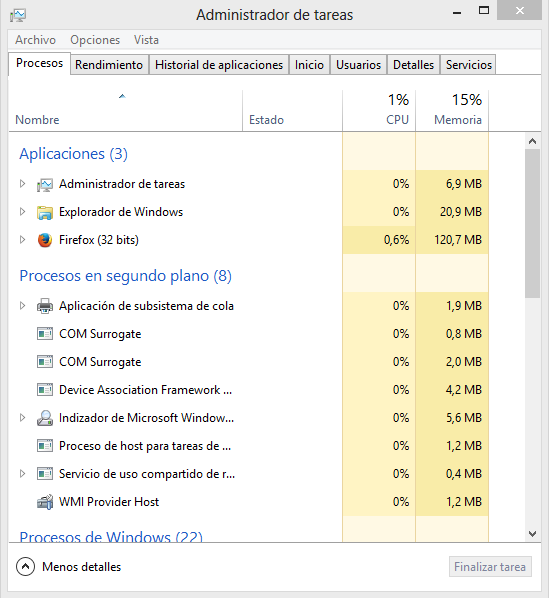
**AngularJS POST**

****

**Figura 49-4.** AngularJS POST, CPU y Memoria RAM

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**EmberJS POST**

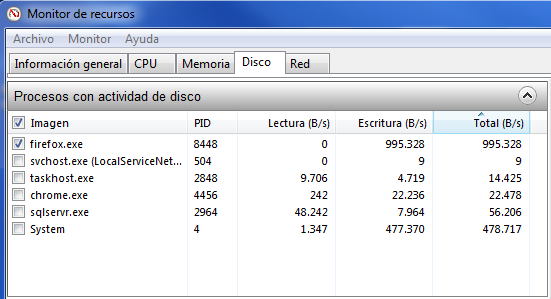
****

**Figura 50-4.** EmberJS POST, CPU y Memoria RAM

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Indicador Disco.**

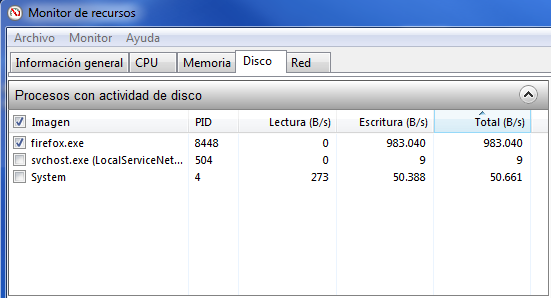
**AngularJS GET**

****

**Figura 51-4.** AngularJS GET, Disco

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

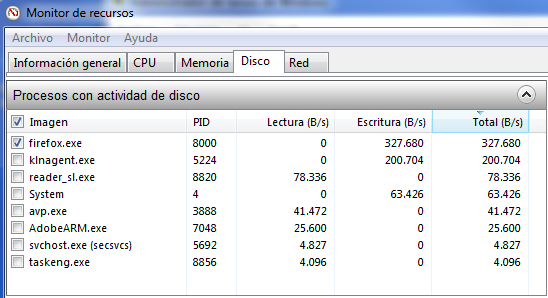
**EmberJS GET**

****

**Figura 52-4.** EmberJS GET, Disco

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

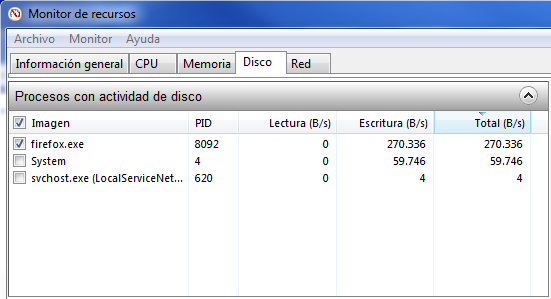
**AngularJS POST**

****

**Figura 53-4.** AngularJS POST, Disco

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**EmberJS POST**

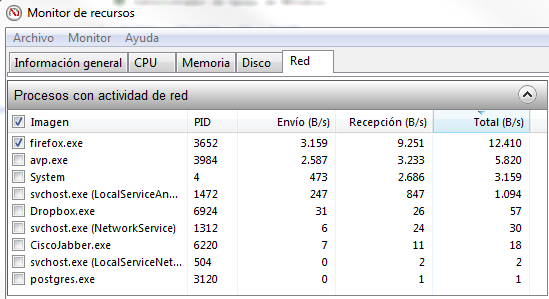
****

**Figura 54-4.** EmberJS POST, Disco

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Indicador Red.**

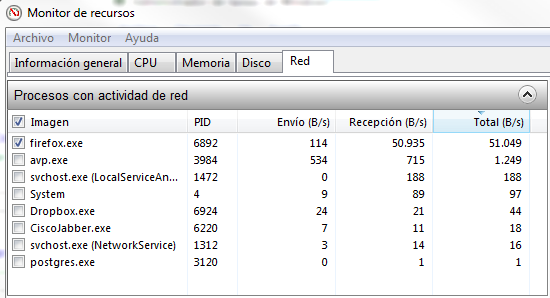
**AngularJS GET**

****

**Figura 55-4.** AngularJS GET, Red

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

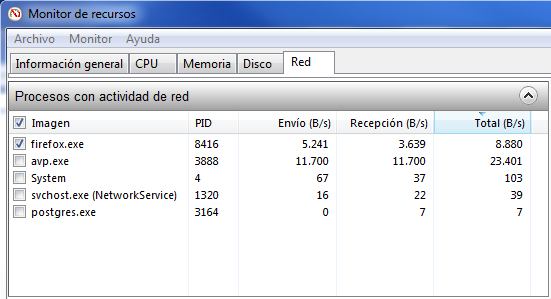
**EmberJS GET**

****

**Figura 56-4.** EmberJS GET, Red

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

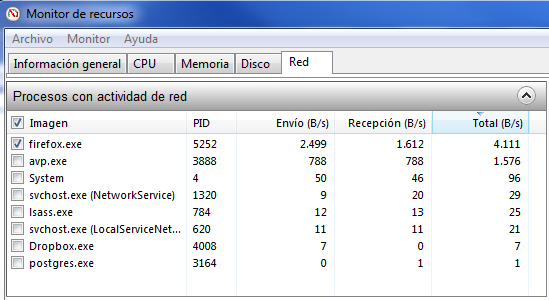
**AngularJS POST**

****

**Figura 57-4.** AngularJS POST, Red

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**EmberJS POST**

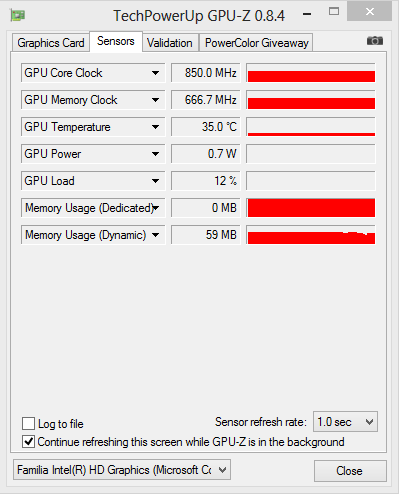
****

**Figura 58-4.** EmberJS POST, Red

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

**Indicador Tarjeta Gráfica.**

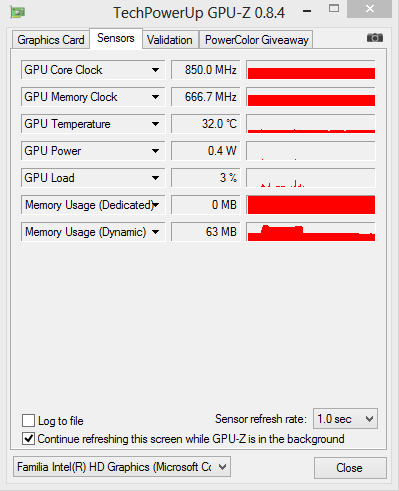
**AngularJS GET**

****

**Figura 59-4.** AngularJS GET, Tarjeta Gráfica

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

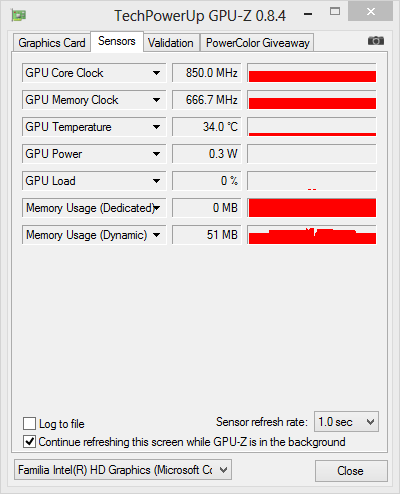
**EmberJS GET**

****

**Figura 60-4.** EmberJS GET, Tarjeta Gráfica

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015

Anexo B: Utilización de Tarjeta Gráfica por el computador, sin ejecución de los prototipos.

****

**Figura 61-4.** Tarjeta Gráfica sin ejecución de prototipos

**Fuente:**  PEÑA, J., CAMBISACA, M., 2015